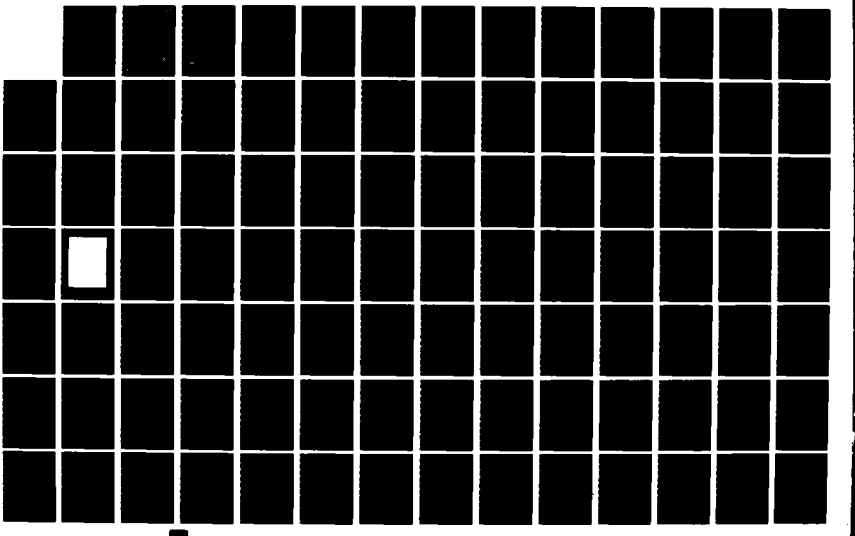
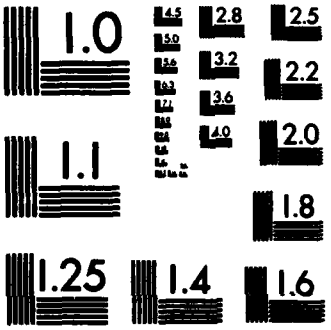
AD-A135 444 QUALITATIVE REASONING IN AN EXPERT SYSTEM FRAMEWORK(U) 1/2
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
S E CROSS MAY 83 AFIT/CI/NR-83-70D

UNCLASSIFIED F/G 5/1 NL





QUALITATIVE REASONING IN AN EXPERT SYSTEM FRAMEWORK

BY

STEPHEN EDWARD CROSS

B.S.E.E., University of Cincinnati, 1974
M.S., Air Force Institute of Technology, 1977

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1983

Urbana, Illinois

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 83-70D	2. GOVT ACCESSION NO AD-A135 444	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Qualitative Reasoning in an Expert System Framework		5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Stephen Edward Cross		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: University of Illinois Urbana-Champaign		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE 1983 13. NUMBER OF PAGES 129
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

29 NOV 83

LYNN E. WOLAVER
Dean for Research and
Professional Development

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

[redacted]

Accession For
NTIS GRA&I [X]
DTIC TAB [ ]
Unannounced [ ]
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A/1

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Researcher: Stephen Edward Cross, Captain, USAF
Title: Qualitative Reasoning in an Expert System Framework
Degree: Doctor of Philosophy
University: University of Illinois at Urbana-Champaign
Year: 1983
Pages: 129

Key Words: Expert Systems, Qualitative Reasoning, Constraint Propagation, Plan Justification, Artificial Intelligence

Abstract

Explanation capabilities of an expert system in the domain of air traffic control is examined. Qualitative reasoning is used to justify heuristically generated plans. The justification is based on mathematical knowledge of aircraft performance which is computationally too complex for use in the normal planning process. Equations are represented in a semantic network which provides a structure in which qualitative values are propagated. The approach is unique in three aspects. First, an abstraction level is used. The equations can be interpreted in terms of Newtonian mechanics as applied to the one dimensional motion. Second, the reasoning process is bidirectional. Third, the computer generates the semantic structure based on a symbolic series expansion of the equations.

Selected Bibliography

[1] Aikins, J. (1980). Prototypes and Production Rules: Knowledge and Representation for Consultation. Ph.D. Dissertation. Stanford University. STAN-CS-80-814.

[2] Clancey, W. (1981). The Epistemology of a Rule-based Expert System: A Framework for Explanation. Stanford University. STAN-CS-81-896.

[3] de Kleer, J. (1979). Causal and Teleological Reasoning in Circuit Recognition. Ph.D. Dissertation. Massachusetts Institute of Technology. MIT-TR-529.

[4] Forbus, K (1982). Qualitative Process Theory. Massachusetts Institute of Technology. MIT-AI-TR-664.

[5] Rieger, C. (1975). The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief, and Contextual Language Comprehension. University of Maryland. TM-373.

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

MAY 1983

WE HEREBY RECOMMEND THAT THE THESIS BY

STEPHEN EDWARD CROSS

ENTITLED QUALITATIVE REASONING IN AN EXPERT SYSTEM FRAMEWORK

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

Director of Thesis Research

Head of Department

Committee on Final Examination†

Chairman

† Required for doctor's degree but not for master's.

O-517

AKNOWLEDGEMENT

I owe a great debt to many.

- To the Air Force Institute of Technology for providing me the opportunity for graduate study,

- To Professor Robert T. Chien, my advisor, who provided the resources for this research and whose advice was always true,

- To Dr. David Waltz, who provided much needed advice and encouragement,

- To Dr. Gerald DeJong who made me the 'hacker' I am today,

- To Shaun Keller, my office mate, who endured my bad moods and crazy ideas, and through our ongoing discussions positively influenced this work,

- To Mrs. Barbara Champagne, who tactfully obtained needed access to the boss,

- To my friends at the Windsor Road Christian Church, whose love and encouragement enabled me to finish this work and served as a constant reminder that my research was not the most important aspect of my life,

- To my wife, Sue, and children, Stacey and Scott, whose love was freely given and who provided stability to my life during this stressful period of graduate study.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Expert systems typically utilize a declarative and uniform knowledge representation (Stefik [60]). The approach offers many operational advantages (e.g., a simple control structure), but is limited to expressing an expert's surface level knowledge in the form of pattern-decision pairs (Chandrasekaran and Mittal [12]). The computer should have access to 'deeper' knowledge if it is to understand and justify its planning actions. Consider a domain where knowledge in the form of equations and algorithms is computationally too complex for use by the human practitioner. How should mathematical knowledge be represented to aid in the improvement and justification of plans? In the task domain of this research, enroute air traffic control, heuristically generated plans are justified by applying qualitative reasoning to aircraft performance equations. Equations are represented in a semantic network where nodes represent variables and links represent dependent variable influences.

The approach is unique in three aspects. First, a level of abstraction is included. Domain equations may be computationally too complex for a human expert to use. However, the equations can be interpreted in terms of a naive representation of Newton's laws as applied to one dimensional motion thus abstracting the influences inherent in the equations. Second, the approach enables bidirectional reasoning. Qualitative knowledge can be used to direct quantitative reasoning. Additionally, when new equations are implemented, their meaning is represented

explicitly and interpreted using the existing qualitative knowledge. Third, the computer constructs its own representation of the equations based on a symbolic series expansion.

1.0. An Example Problem

Consider two aircraft that are involved in a 'head-on' conflict. The controller must generate a plan that prevents a midair collision. The plan must involve the modification of one of the aircraft's flight plans. If collision avoidance were the only air traffic control goal, the solution would be trivial. Any legal operator (e.g., climb, descend) could be used. However, there are other important goals such as fuel efficiency. A significant portion of the controller's training involves assimilating heuristics useful for generating plans that achieve both goals. For instance, aircraft are usually more fuel efficient at higher altitudes. The human controller chooses an operator that prevents a collision and improves (or at least does not seriously degrade) fuel efficiency.

Many of the heuristics that controllers use are justified by mathematical knowledge. Assume that the controller requires one of the aircraft to climb, thus achieving the required separation. The climb was chosen because the controller recognizes that aircraft are more fuel efficient at higher altitudes. The declarative statement that 'aircraft are fuel efficient at higher altitudes' requires many equations for justification. Fuel efficiency is dependent on the fuel flow rate, which in turn is dependent on the required engine thrust. The required engine thrust is dependent on aircraft drag. Aircraft drag is dependent on aircraft geometry, airspeed, and air density. Justification is predicated

on the computer's understanding of these equations and their relationship to heuristics.

2.0. An Expert System Approach

A problem is defined using aircraft flight plan data which is identical to the initial data the human controller receives. The data is processed and a semantic structure which represents the global aircraft conflict scenario is created. Each node of the structure represents an aircraft and each link represents the type of conflict. The structure is decomposed into individual problems by applying problem decomposition strategies. Some strategies recognize goal interactions. For instance, an aircraft may be involved in recurring similar conflicts with many other aircraft. Each instantiated problem decomposition strategy in turn causes a problem solving strategy to be instantiated. A problem solving strategy specifies the knowledge that allows the computer to select which aircraft should receive a command and what type of commands should be used. The detailed knowledge about commands resides in tactics. Commands are criticized to insure they are do not exceed aircraft performance or cause additional conflicts. The approach is modeled after human controller problem solving behavior. Knowledge is represented in frames. Frames provide the structure in which the diverse knowledge of this task domain can be integrated. A frame language patterned after FRL (Roberts and Goldstein [52]) was implemented in Franz Lisp on a VAX 780. The purpose of this research was to enable the computer to reason about its heuristic knowledge.

3.0. Reasoning Component

The reasoning component consists of naive physics knowledge, domain equations, and an interface between these knowledge types and a data base of heuristics.

3.1. Naive Physics Level

Newton's laws are represented in a semantic network where nodes are primitive concepts (e.g., force, velocity) and links indicate their interaction. The structure represents the designer's understanding of Newtonian mechanics as abstracted to one dimensional motion of mechanical systems. The naive physics knowledge is made explicit by the designer. The linkages between the naive physics level and a domain are equation dependent. The representation serves as the basis for the interpretation of detailed domain equations and algorithms.

Nodes represent forces (propulsive, enabling, resistive), acceleration, velocity, position, mass, and power. Propulsive and enabling forces are referred to as positive forces. Propulsive forces require an external enablement which converts fuel (portion of system's mass) into the force. The rate of change of mass varies directly with the change in propulsive force. That is, when propulsive force increases, the fuel flow rate increases causing the mass rate of change to increase.

Another positive force is called an enabling force. Enabling forces do not directly influence mass. For instance, the air flow over a wing causes a pressure differential that enables lift. Lift is an enabling force that counteracts weight. Resistive forces counteract positive forces. Common examples are pressure and friction forces which vary

with velocity and domain dependent variables such as weight, density, and surface area.

Nodes are related by links which specify how a dependent variable is 'influenced' by a changing independent variable. The links define a structure in which qualitative values are propagated. There are four types of links: influence, component, parent, and instance. Influence links are labeled positive or negative depending on a node's incremental affect on another node. Component links partition equations into terms. Parent and instance links indicate domain and naive physics relationships.

3.2. Domain Equations

Aircraft equations of motion are dependent on four forces: lift (L), thrust (T), drag (D), and weight (W). For level flight, dynamic equilibrium is defined as:

$$L - W = 0 \tag{1.1}$$

$$T - D = 0 \tag{1.2}$$

Each force is defined by an equation. Thrust is a function of throttle setting. Lift is a function of velocity, air density, angle of attack, and wing geometry. Weight is a function of aircraft mass and gravity. Drag is a function of air density, velocity, and aircraft configuration variables. Drag has two components: parasitic and induced drag. All subsonic aircraft performance capabilities can be derived from the drag equation (1.3). Consider an aircraft at a constant

altitude and configuration. The maximum velocity then occurs when the drag equals the maximum thrust. Since drag is parabolic with velocity, the velocity at the minimum drag defines the 'best endurance' airspeed.

$$D = D_p + D_i \tag{1.3}$$

$$D_p = \frac{f\, V^2\, \sigma}{295} \tag{1.4}$$

$$D_i = \frac{94}{\sigma e} \left|\frac{w}{b}\right|^2 V^{-2} \tag{1.5}$$

where,

v = velocity
w = weight
σ = altitude density ratio, and
b, e, and f are aircraft configuration variables

A symbolic series expansion is used to define the influence links. The sign of the first error term indicates a positive or negative influence. The magnitude of the influence is the amount of the first error term and is saved if ambiguity resolution is later required. There are four types of influence links: primary positive, primary negative, secondary positive, and secondary negative. The primary/secondary distinction is required for search efficiency. For instance, acceleration is primarily influenced by force and secondarily influenced by mass.[1] A variable is defined as a primary influence if it is an instance of an abstracted concept and that concept is also a primary influence. Air-

[1]This is the author's interpretation of the physics. Acceleration usually changes in dynamic systems because one changes the applied force, although there are instances when the mass is changed (e.g., discarding ballast in hot air balloons).

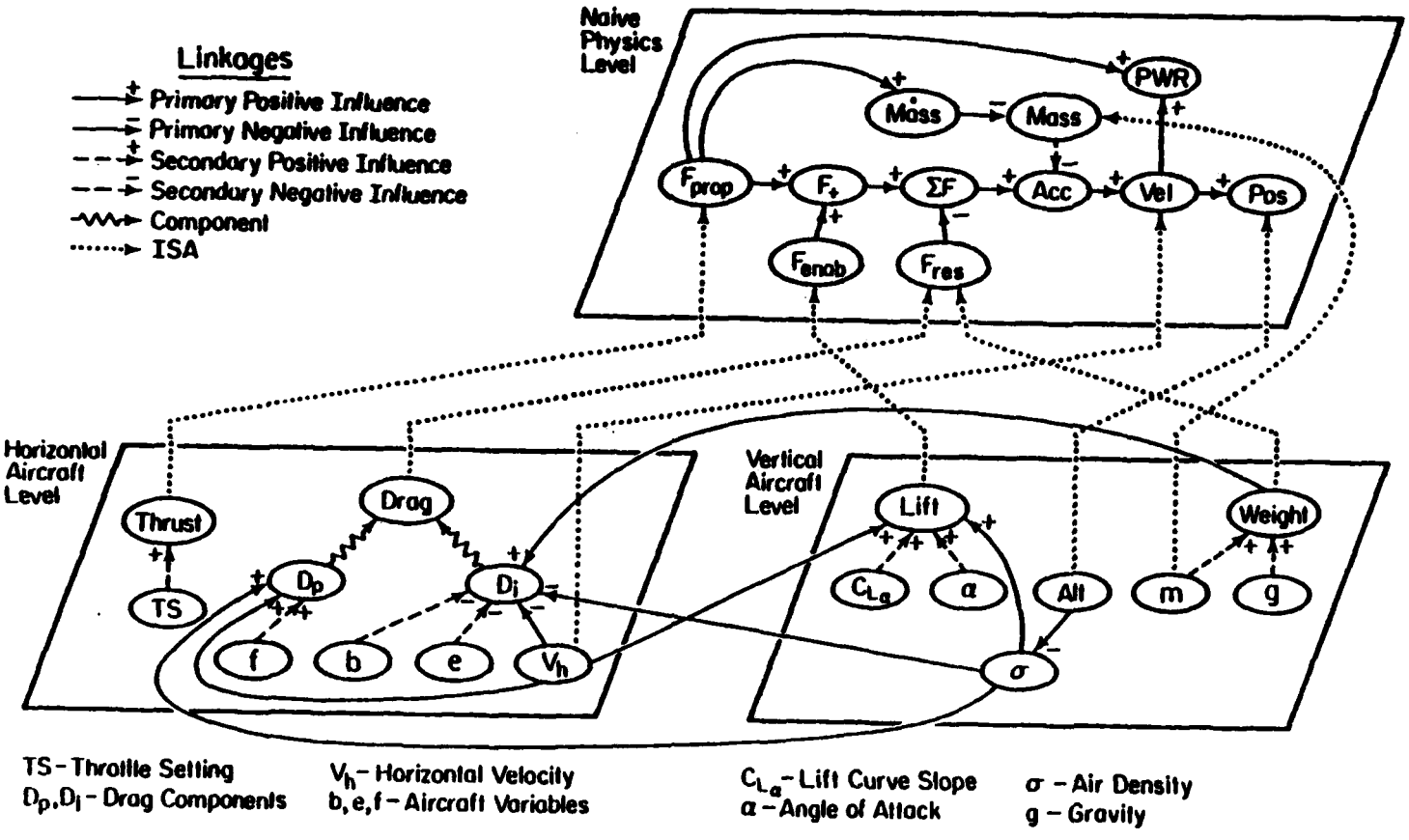
craft airspeed (or horizontal velocity) is a primary influence because it is an instance of velocity which is itself a primary influence of position. Sometimes influences are found recursively. For instance, air density is a function of altitude (an instance of vertical position). Position can be a primary influence of resistive force (e.g., friction). Thus, it is inferred that air density is a primary influence of drag.

3.3. Interfaces

The signs of the terms in the force equations (1.1) and (1.2) are used to define instances of forces in the naive physics representation. For instance, T and L are positive forces. Semantic knowledge is also required. Thrust is a propulsive force while lift is an enabling force. In this context, weight and drag are resistive forces. The representation of both aircraft levels and the abstracted structure is shown in Fig. 1 (p. 8). Conceptual knowledge specifies the procedures to use in different contexts. A plan is fuel efficient if after implementing the plan, fuel consumption decreases. Less fuel is used if the drag decreases since thrust equals drag in level flight. Consider the case where a climb command is given. From the structure of Fig. 1 (p. 8), it is seen that the air density term decreases. This may or may not cause a favorable change in drag. The effect of error terms on the drag components due to changes in density are retrieved and the pertinent computation is performed.

4.0. Applications of the Research

Plan justification enables computer understanding and improved explanation capabilities. The justification for the climb command is



Interface to the Aircraft Levels
Figure 1

that it prevents a midair collision and is fuel efficient. The important point is that the computer understands the fuel efficient aspect in terms of the equations and their semantic structure. It can explain its justification at a more abstract level. The decrease in air density causes drag to decrease. Drag is a resistive force. Since resistive force decreases, the positive forces can also decrease.

The diagram shown in Fig. 1 (p. 8) is useful for common sense reasoning about aircraft performance. A controller should never issue a command that cannot be implemented by an aircraft. Assume an aircraft is commanded to increase its speed. Implicit in the command is constraint on altitude (i.e., constant). Can an aircraft increase its speed without increasing its altitude? The answer is obtained by a combination of forward and backward constraint propagation in the naive physics and domain equation representation. By applying conceptual knowledge about aircraft (e.g., wing size is constant, lift curve slope changes only when the flaps are deployed) the computer reasons that (1) the throttle setting is increased resulting in the increased speed and (2) the angle of attack is decreased resulting in a constant altitude.

The approach is useful in the learning of new problem solving strategies. Human controllers acquire their skill by the justification and assimilation of an expert controller's plans. I say that this type of learning is advice-initiated. A prerequisite of advice-initiated learning is that the computer understand the advice. I say that this process is advice interpretation. I define advice interpretation as the justification of another expert's plan. For instance, a new control procedure requires arrival aircraft to be descended 50 miles earlier

than was previously authorized (Stengel and Marcus [61]). The reasoning procedure must be applied to the entire flight path to justify the reasons for overall decreased fuel consumption.

5.0. Related Work and Open Research Problems

Several recent expert systems have used a frame representation (Aikins [1], Stefik [59]). Frames provide a structure in which context related knowledge can be integrated. Knowledge organization is a necessary, but not sufficient, step towards computer understanding of planning actions. In this research, I use a frame representation and show how the computer can reason about the heuristic knowledge contained in frames.

Hayes [33] discussed the need for naive theories and provided a theoretical framework for future work. de Kleer [25] explored the computational aspects of qualitative reasoning in the mini-world of the roller coaster and contributed the concept of envisionment. Envisionment predicts system behavior through qualitative simulation. In related research [26], he illustrates the use of Incremental Qualitative (IQ) analysis as a weak form of reasoning about perturbation. Forbus [31] uses a stronger approach that includes the sign and magnitude of a quantity's amount and derivative.

Qualitative reasoning is used to justify heuristically generated plans with knowledge that is computationally complex. The approach is motivated by conceptual dependency (Schank and Abelson [55]) where any implicit knowledge in an input is made explicit in an internal representation. Equations are represented in a semantic network which is sugges-

tive of Rieger's common sense algorithms [51], but is purposely less expressive in the types of linkages.

The thesis of this research is that mathematical knowledge can and should be used in expert systems. A qualitative reasoning capability facilitates the integration of mathematical knowledge with a heuristic knowledge base.

6.0. Overview of the Dissertation

Chapter II reviews the state of the art in expert system knowledge comprehension. The task domain of enroute air traffic control is described in Chapter III. The expert system architecture and frame representation language are discussed in Chapter IV. In Chapter V, the qualitative reasoning approach is developed. Applications to plan justification and advice interpretation are presented in Chapter VI.

CHAPTER II

SUMMARY OF RELATED WORK AND OPEN RESEARCH PROBLEMS

An expert system that functions in a real world domain not only requires an extensive knowledge base, but requires the capability to reason about its knowledge and to acquire new knowledge. In this dissertation, I show that an expert system can justify plans based on knowledge that is computationally too complex to be used in the 'normal' planning process. By normal, I mean the use of heuristic knowledge to guide the search for plans that achieve explicit goal statements. The heuristic knowledge base is interfaced to a mathematical knowledge base. Mathematical knowledge is made explicit and transparent to a qualitative reasoning process that interprets equations and algorithms in terms of semantically relevant domain concepts and heuristics. In this chapter, I review the current state of the art of expert systems. I concentrate on the knowledge understanding problem and relate recent research in qualitative reasoning. The applicability to knowledge-based learning research is discussed. I conclude with a description of the research problems addressed in this dissertation.

1.0. Overview of Expert Systems

Expert systems are often identified with the production system paradigm. Production systems can be traced to the early work of Post [49] and are now used in a wide range of applications, from psychological experiment modeling (Anderson [4]) to expert system applications. Examples of expert system domains include medicine (Shortliffe [56]), computer configuration (McDermott [42]), investment advice (Davis [20]),

genetic experiment design (Stefik [59]), and mineral exploration (Duda et al. [28]).

A production system has three components: a knowledge base, a data base, and a control structure. In the purest form, the data base is a centralized storage medium of primitive symbols. All knowledge is encoded in a uniform knowledge representation (called production rules or rules) and is stored in a central location called the knowledge base. Rules are antecedent-consequent pairs. If the symbols in the antecedent of a rule match the data base, then the symbols in the consequent can be written into the data base. The control structure cycles through the rule base until a matching rule is found, allows execution of that rule, and continues cycling through the knowledge base.

All production systems follow this general outline but in practice are more complicated. Rather than simple pattern matching, a rule may contain predicate functions that perform operations on the data base. Each rule is a unique encoding of a 'chunk' of knowledge. In theory, the knowledge base is unordered and the rules are independent of one another. However, in practice it is quite common to hand order rules and encode specialized 'message passing' rules.

The control structure operates in a select-and-execute fashion. By applying a problem solving strategy, a rule is chosen for execution and the results of that rule affect the data base. The problem solving strategy operates by forward chaining or backward chaining. In forward chaining, the antecedents are compared or evaluated with respect to the data base until an applicable rule is found. The rule is executed and the search continues. DENDRAL (Feigenbaum et al. [30]) is an example of

a production system that uses forward chaining. A different approach is backward chaining. Given a goal statement, a search is made for all rules that conclude something about the goal. Subgoals are thus formed and the search for applicable rules continues recursively. MYCIN (Shortliffe [56]) is an example expert system that uses backward chaining.

If more than one rule are found, there is a conflict. That is, the control structure must decide which one of the many applicable rules should be applied. McDermott and Forgy [43] describe the domain independent methods that are typically used to resolve these conflicts. For instance, the computer may choose the most recently acquired rule, the historically most successful rule, or the 'least cost' rule.

The production system approach has several advantages. The knowledge is acquired from experts in the form of 'condition/action' heuristics and production rules provide a natural representation. Since the control structure is separated from the knowledge base, rules can be deleted, modified, or added without affecting system operation. The rules are logical constructs and solutions can be proven to be logically correct with respect to the knowledge in the knowledge base. Correctness has been an overriding concern in previous expert system research. The 'proof' serves as an explanation to the human user. This is especially useful in the performance testing phase. The production system answers question like 'WHY' (why a question was asked by the computer) or 'HOW' (how a conclusion was reached) by reciting some portion of the rule chain that was used to achieve a goal. The advantages of the production system approach are chiefly implementational. I will now discuss a disadvantage, their lack of domain understanding.

Production systems do not understand their domains. Each rule contains a microscopic 'chunk' of knowledge that does not relate to the 'macroscopic' context of a domain. There is no convincing argument that humans represent their knowledge in production rules or that the control structures of forward/backward chaining are typical of human reasoning (Simon [57]). The performance of an expert system might be increased by including knowledge that a human would not normally use. For instance, one may wish to use detailed control algorithms that are computationally too complex for a human. One can imagine such an algorithm's possible, but cumbersome, representation in rule form. A third disadvantage concerns the domain of application. Many domains, such as natural language, are much broader and do not require as deep as inferencing as is usually provided by a production system approach.

Research in language comprehension has addressed similar knowledge organization issues (Charniak [13]). Knowledge is represented in stereotypical structures called frames (or schemas, or scripts). These structures offer a natural way to partition knowledge. The structures are used to understand pieces of text. That is, the structure specifies the common-sense knowledge that is required to interpret the text. If one wished to interpret 'What color is a pear?' one would not retrieve facts like 'fire hydrants are red' (Charniak [13:227]). One of the major tenets of conceptual dependency theory (Schank and Abelson, [55]) is that any information in a sentence that is implicit must be made explicit in the computer's meaning representation of that sentence.

The explanation capabilities of production systems are often confused with understanding (which is the responsibility of the human

user). The usefulness of the automated explanations is dependent on the ability of the user to understand the chaining among rules and on the content and clarity of the knowledge represented in the rules. These two assumptions are not always met. Clancey [18] describes how the rules of MYCIN do not capture the human expert's understanding of rules. For instance, a rule like 'If the patient is less than eight years old, do not administer tetracycline' does not represent the causal knowledge that tetracycline can impair bone development in children. Aikins [1] has investigated the knowledge organization question for medical based expert systems while Pople [48] has sought to implement a more natural reasoning capability. Both have represented knowledge in frames and have achieved performance more consistent and easily understood by an expert diagnostician.

I have adopted the frame as the knowledge representation used in the expert system design. I use frames to integrate diverse knowledge representations (e.g., rules, equations). Their usage is as a representation that allows the categorization of different types of rules. In this sense, frames serve much the same purpose as Davis' meta-rules.

A frame representation is a necessary, but not sufficient, step towards computer understanding of planning in a task domain. The sufficiency argument invokes the 'frame hypothesis' (Charniak [14]) which states that understanding an input is equivalent to finding a frame in which the input can be integrated. There are at least two problems with the hypothesis.

1. Often there is no one frame into which inputs can be integrated. There are two reasons for this. First, an applicable frame may not exist

which implies that new knowledge need be acquired. Second, the input may contain a goal interaction which cannot be handled by the present system of frames.

2. There may be a number of frames into which inputs can be integrated, but doing so is not tantamount to understanding. Consider the statement 'He painted the terminal screen white' (Charniak [14]). If one only had the normal painting frame, the input sentence could be interpreted. But the computer also needs the ability to justify its interpretation. In this case, the justification would be based on world knowledge of 'painting' and 'terminal screens.' In the domain of air traffic control, at least part of the justification can be made based on the computer's understanding of the aircraft equations of motion.

2.0. Qualitative Reasoning

Hayes [33] first described the need for the use of 'naive physics' theories in artificial intelligence systems. McCloskey [41] recently reviewed the psychological literature and his own experiments describing the naive theories of motion. It is interesting that most people have a Galilean concept of motion until they are taught formal physics. Often, the naive views are an impediment to comprehension of Newtonian physics. de Kleer [25] explored the computational issues in a problem solver in the world of roller coasters. Two knowledge representations were used. One was based on declarative, qualitative statements such as the general shape of curves or the relative heights of points. The other was based on quantitative knowledge about specific domain equations. His contributions was a weak form of physical reasoning called envisionment which allowed the computer to predict the future states based on qualitative

knowledge. Bundy [10] used a common reasoning approach to accomplish the same purpose. Recently, Forbus [31] has extended the work of de Kleer to include knowledge about the sign and magnitude of a quantity's amount and derivative. Rieger's [51] Common Sense Algorithms (CSA) provide a language for the description a physical process. The resulting structure is a semantic network which can be used for qualitative simulation.

I am motivated by the research of de Kleer, Forbus, and Rieger. My contribution is twofold. First, I take advantage of the structure of mathematical statements to allow the computer to construct its own semantic network representation. Nodes represent variables and the links between nodes are derived from a symbolic series expansion of an equation. The representation is interfaced to abstract concepts of force and motion thus providing a capability to reason about Newton's laws. The meaning of domain dependent mathematics, which would be computationally too difficult to use in a heuristically based problem solver, can be made transparent and explicit to a conceptual knowledge base. The primary use of the capability will be in the justification of plans. This will be shown to also be useful in the advice interpretation phase of learning.

3.0. Automated Learning

Inductive learning is the most often cited approach to automated learning. Induction requires one to:

(1) Define a training set of positive and negative instances and

(2) Propose target concepts that retain all of the features of the positive instances and none of the features of the negative instances.

The approach has been applied to many expert systems. Meta-DENDRAL (Buchanan and Mitchell [9]) discovered useful rules for the interpretation of mass spectroscopy data. Mitchell [45] improved the search algorithms in a program that implemented his version space theory. Winston's transfer frame approach implicitly defined the feature space as the slots of the subject and target frames [67]. There are several points which restrict induction as an autonomous approach to machine learning.

(1) A human expert specifies the feature space (implicitly restricting the concepts that can be learned),

(2) A human expert decides which learned concepts are indeed useful, and

(3) A human expert labels each training instance as positive or negative.

The last three statements are significant because the deficiency of the inductive approach is not in the approach itself, but in the learning system's lack of domain understanding about what is already known and its relation to what is to be learned.

Advice-initiated learning is a knowledge-based learning approach. Examples of previous knowledge-based learning research are Soloway's BASEBALL program [58] and DeJong's Explanatory Schema Acquisition theory [24]. BASEBALL learned rules of baseball by generalizing its observations from actual games. It used its heuristic knowledge of competitive games to propose the features upon which a generalization should be made. That is, the computer used its own knowledge to construct a feature space with which inductive-like learning was accomplished. DeJong's Explanatory Schema Acquisition approach seeks to construct generalized scripts from perceived novelties in stories. The automated

recognition of air traffic control 'novelties' is a difficult problem compounded by lack of a structured domain theory. I allow the novelties of air traffic control to be expressed by an outside expert's 'hint'. The computer interprets the hint with respect to its previously encoded strategies and tactics and then constructs an appropriate representation for future problem solving. The approach is consistent with the training environment of developmentals in the DYSIM.

McCarthy's 'Advice Taker' was an early attempt at advice-initiated learning [40]. Since that time the research has followed two paths. The first area is concerned with the construction of sophisticated tools for the construction and maintenance of knowledge bases (Davis [21]). The second area has dealt with translation of high-level, abstract, and often ambiguous statements of advice into representations that are usable by the computer. Mostow's FOO program [46] was used to investigate the capabilities that a computer must have to accomplish advice-initiated learning. These capabilities included advice interpretation, operationalization, utilization, and generalization. Advice interpretation concerns itself with translating a statement of advice into an explicit, unambiguous expression. In this sense, the translation of a story into conceptual dependency could be considered an advice interpreter. Mostow concentrated on the operationalization problem and described the types of rule-based knowledge that would have to be used to transform an unambiguous advice statement into an executable procedure.

4.0. Open Research Problem

Production rules do not, in practice, capture an expert's understanding of his domain, and explanations constructed by the recitation of a rule-based reasoning chain are often unsatisfactory. New knowledge needs to be provided to the system but the computer also needs to be able to understand the intended uses of that knowledge. Heuristic knowledge organized in frames is a necessary condition for computer understanding. A central problem in automated understanding is the computer's inability to <u>justify</u> its actions. The purpose of this work is to show that an expert system can justify its plans based on knowledge that is computationally too complex to use in the normal planning process. The same reasoning process is shown to be useful in the interpretation of ambiguous advice. Specifically, I show the <u>feasibility</u> of integrating a conceptual knowledge base with a mathematical knowledge base. The interface is based on a qualitative representation of Newtonian physics as applied to one-dimensional motion.

CHAPTER III

A TASK DOMAIN: ENROUTE AIR TRAFFIC CONTROL

In this chapter, the application domain of high altitude enroute air traffic control is described. This is an interesting domain for several reasons. First, the task of air traffic control is highly dependent on the skill of human experts and acquisition of such knowledge for computer representation is a challenging problem. Second, the Federal Aviation Administration has decreed that the task be automated (Lerner [39]). There is a large body of algorithmic knowledge available which can be applied to portions of the task. I concentrate on the mathematical descriptions of aircraft performance. An expert system that performs the air traffic control task will need to be able to solve problems and justify solutions based on its understanding of both types of knowledge. The domain will first be described, followed by a description of the human controller training process, his knowledge, and planning behavior. A short discussion of mathematical knowledge will then be given.

1.0. Geographical Description

The United States consists of 23 Air Route Traffic Control Centers (ARTCC). An ARTCC is divided vertically into terminal, low-altitude enroute, and high-altitude enroute control sectors. Each sector is manned by a controller whose goal is to insure conflict-free, expeditious transit for each aircraft in the sector. There is a strong bias towards safety as evidenced by the introductory comments in the Air Traffic Control Handbook [3:7]

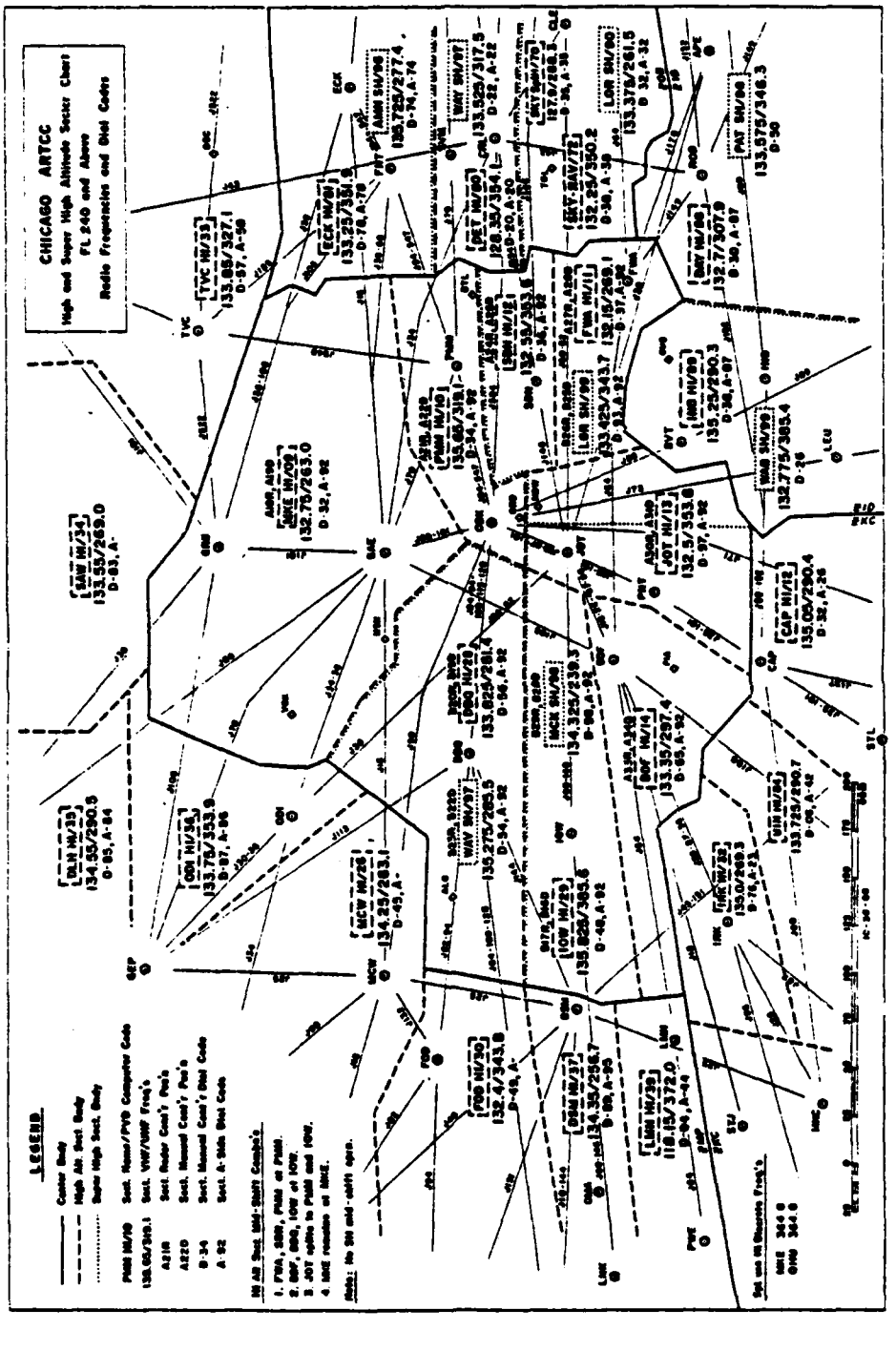
> Give first priority to separating aircraft and issuing safety advisories as required in this handbook. Good judgement shall be used in prioritizing all other provisions of this handbook based on the requirements of the situation at hand.

Controllers are subject to constraints from other sectors and supervisory controllers (e.g., flow control constraint), but are free to use one of five operators to resolve a conflict. These operators consist of aircraft turns, climbs, descents, speed changes and holding patterns. Aircraft have one of three intents: arrival (will descend to a nearby airport), departure (entering the enroute sector from a nearby airport) or overflight (an enroute aircraft in level flight or altitude transition).

For the purposes of this research, a simulation program was created that modeled the eight high altitude sectors of the Chicago ARTCC (Fig. 2, p. 24). The map data base for the simulation consists of the 8 sector boundaries, 52 airways, and 112 navigation aids (e.g., VORTACS, TACANS).

2.0. Controller Training

Human controllers learn rules, constraints, phraseology and elementary problem-solving strategies during their initial training at the FAA Academy. Subsequent on-the-job and simulator training conducted at their respective ARTCCs focus on the development of conceptual knowledge. The training problems from the Chicago ARTCC simulation facility form the basis of the simulation input. The problems used are based on the training descriptions for an entry level radar controller during his advanced instruction (a person with about three years of experience). A typical training problem consists of an hour of flight

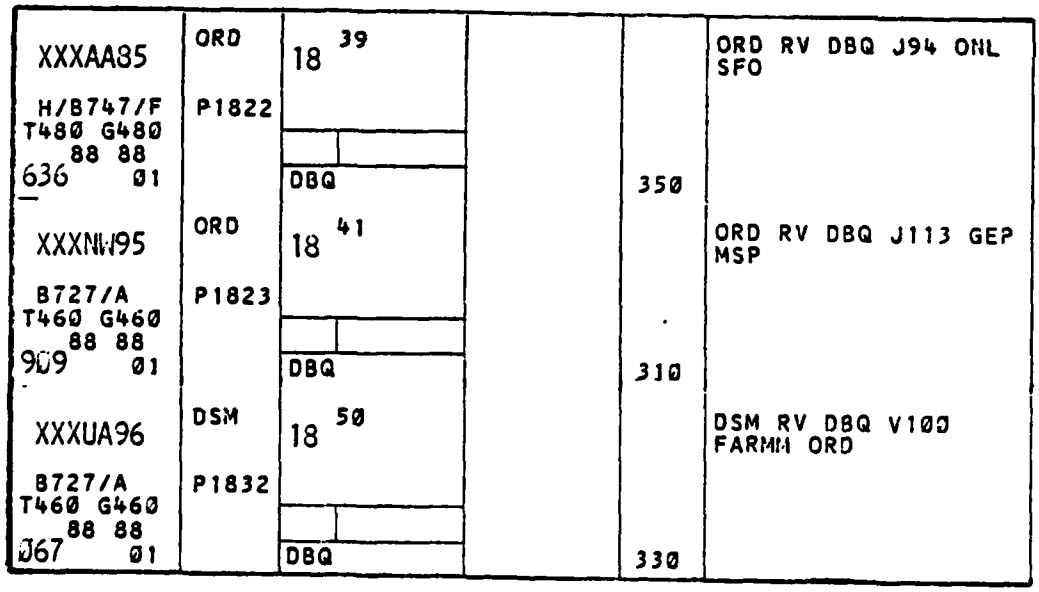


Chicago ARTCC
Figure 2

time, thirty aircraft, and fifteen conflict situations. An instructor controller may ask the trainee to justify his solutions or offer advice which may result in a better plan. A problem is presented in the form of flight strips, three of which are shown in Fig. 3.

3.0. Domain Knowledge

I have acquired the domain level knowledge for the conflict resolution problem in high-altitude enroute air traffic control through discussion with controllers and review of several technical manuals [2, 3, 29]. The knowledge is classified by controllers as strategic and tactical knowledge. By strategic knowledge, I mean the heuristics the



Aircraft Flight Strips
Figure 3

controller used to recognize conflicts, to select aircraft subject to resolution commands and to propose plans. Plans consist of verbal commands that are manually relayed to aircraft and result in flight path modifications. These decisions are functions of the conflict type (crossing, headon, merging or overtake), the aircraft intent (arrival, departure, enroute-level, enroute-descending or enroute-climbing) and the relative state vector. By tactical knowledge, I mean the detailed rules, constraints and algorithms used to accomplish planning for a specified goal. Proposed commands are criticized at both levels. At the tactical level, the controller insures that implementation of a command does not cause an aircraft to exceed its performance capabilities. At the strategy level, the evolving composite plan is criticized to insure invocation of a plan does not cause additional conflicts.

4.0. Human Controller Planning Behavior

Controllers perceive conflict situations fifteen to twenty minutes in advance, and typically issue resolution commands three to five minutes in advance. The commands are chosen to achieve the high level goals of safety and expediency, and form part of an evolving 'bug prone' plan. That is, each command is chosen based on stereotypical knowledge of past conflict situations. A proposed command is criticized by comparing it against known constraints, such as not causing another conflict during a brief time window.

The controller's stereotypical knowledge is obtained during a lengthy training period. The initial training at the academy concentrates on the acquisition of domain rules and constraints. The trainee learns 'by being told.' After being assigned to a control center, the

trainee is apprenticed to a journeyman controller. He learns through on-the-job training and by solving training problems in the Dynamic Simulator (DYSIM). The later form of learning involves understanding and representing the advice of another expert. A few observations are significant.

1. Controllers recognize elegant solutions to conflict resolution problems. In fact, they learn by the incremental acquisition of those solutions.

2. Elegant solutions are characterized by a controller's ability to localize conflicts, relax constraints, and recognize and plan for multiple goals.

a. Conflict localization is the ability to rapidly digest a traffic scenario and predict the conflicts. The controller knows which conflict pairs are related either spatially or by the similarity of the commands he will invoke.

b. Constraint relaxation refers to situations when a controller elects to 'break' the rules. For instance, a controller may elect to violate an aircraft's protection circle.

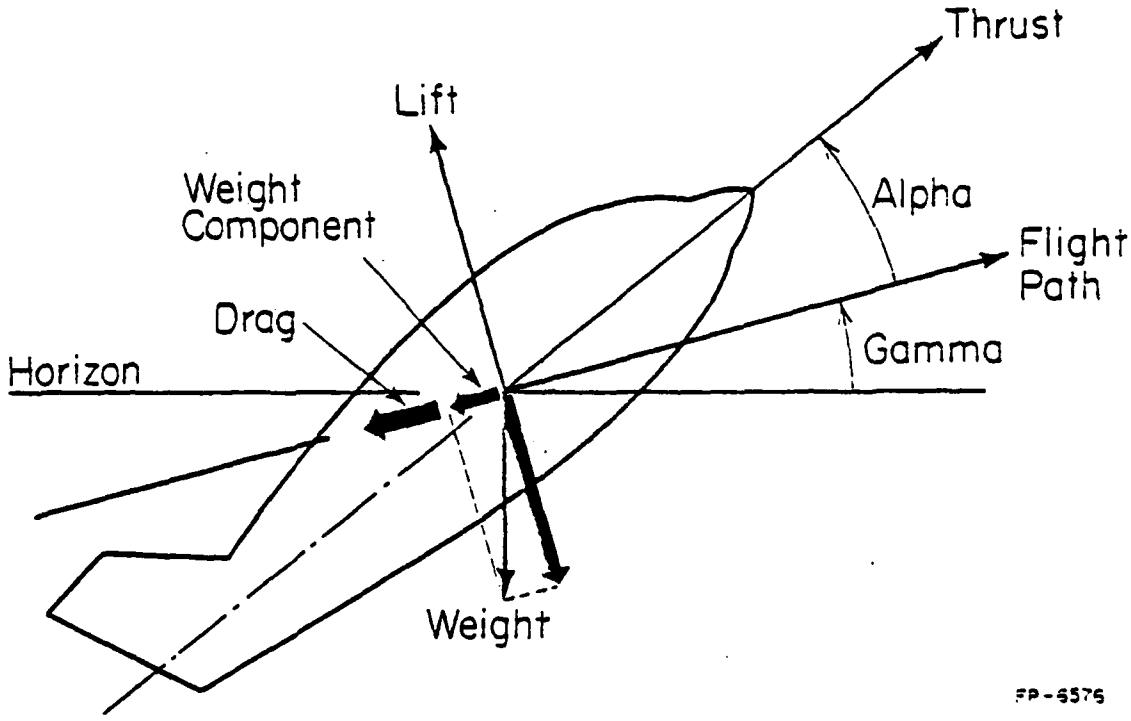
c. Multiple goal satisfaction refers to the controllers ability to utilize a reduced command set to realize several goals. Examples include resolution of several conflicts with one command or achieving a fuel efficient resolution in a novel way.

3. Controllers comprehend the plans of other controllers even though they evolve different styles of controlling aircraft. Their styles reflect the differences in their experiential knowledge bases.

Even though there are differences in their heuristic knowledge of strategies and tactics, it is apparent that a controller understands another controller's plan. Comprehension is demonstrated by the brief descriptions that suffice when one controller hands off his position to another controller. This suggests a controller has a 'deeper' level of knowledge that can be used to justify plans.

5.0. Mathematical Knowledge

Human controllers have a conceptual representation of aircraft performance capabilities. For example, one knows that a fighter jet flies faster than a heavy commercial aircraft or that jet aircraft are more fuel efficient at higher altitudes. The conceptual knowledge is useful in constructing and understanding plans that seek to satisfy nonseparation goals (i.e., fuel efficiency, minimal delay, etc.). There is a mathematical basis for the conceptual knowledge. Consider that the aircraft shown in Fig. 4 (p. 29) is influenced by four forces: lift, drag, thrust, and weight. Significantly, the drag equation (partially a function of lift) is sufficient to compute all subsonic aircraft performance capabilities. The aerodynamic engineer refers to the drag equation as the 'drag polar' since for any given altitude, drag is parabolic with respect to velocity. The maximum velocity of an aircraft is defined as the velocity causing a drag equal to the maximum available thrust. The climb capability is defined by the difference between available thrust and drag. The velocity for maximum fuel efficiency is at the minimum of the drag polar. The equations are derived and discussed in detail in Appendix A. The equations are used to justify plans which are constructed with conceptual knowledge. For instance, a justification for



Aircraft Force Relationships
Figure 4

increasing and aircraft's altitude to improve its fuel efficiency is dependent on the the air density term in the drag equation and the equation for air density.

CHAPTER IV

THE AIR TRAFFIC CONTROL EXPERT SYSTEM

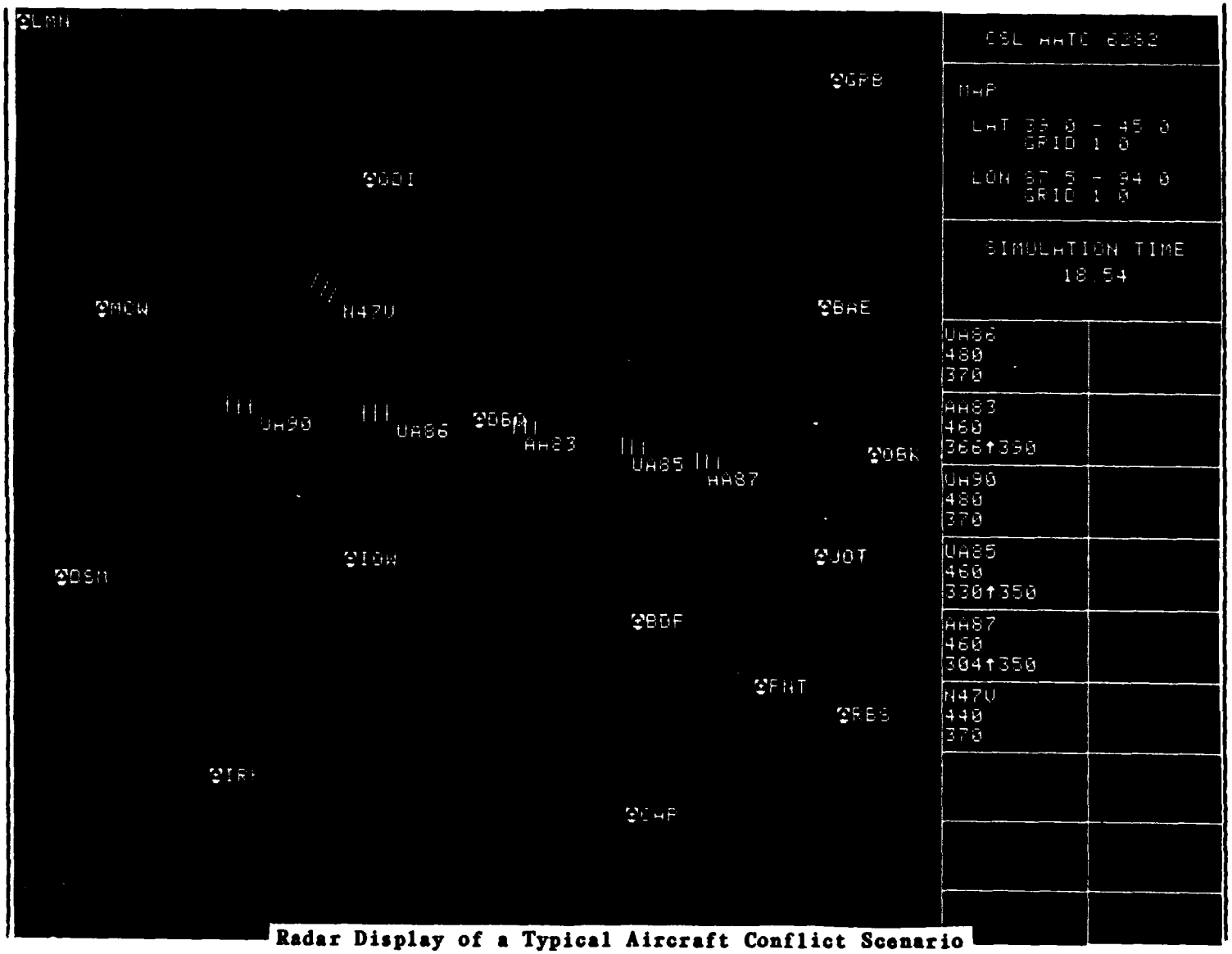
An automated air traffic controller should be characterized by the attributes of elegance previously discussed: conflict localization, constraint relaxation, and multiple goal satisfaction. In this chapter, an expert system architecture is described that will converge to these measures of elegance as a result of the chosen knowledge representation, the problem definition, decomposition, and resolution strategies, and the ability to reason about plans and acquire new knowledge. The capability for understanding domain relevant features of these attributes must be designed into the system. For instance, the expert system defines local conflict aircraft problems using a conflict filter described in Appendix B. The knowledge about how to plan and control air traffic is encoded in stereotypical knowledge structures which controllers refer to as strategies and tactics. Local problems are understood in the context defined by the strategies. Some strategies recognize multiple goals that exist within a global problem statement. I have concentrated on the two primary goals in aircraft control: collision avoidance (a safety goal) and fuel efficiency (an expediency goal). The interactions of these two goals are sufficiently rich so as to make the construction of the expert system challenging. In this chapter, I describe the framework of the expert system design. I begin with a description of the problem definition, decomposition, and resolution strategies. I then discuss their representation in a frame language of my own design and

illustrate a problem solution.

1.0. Preprocessing of Flight Strips

Visual perception is an intangible aspect of the human controller's skill. He can perceive possible collisions from a display of many aircraft. The display may be a time ordered list of flight strips or a computer generated display of radar and transponder data. A simulated version of the radar display is shown in Fig. 5 (p. 33). The approach taken in this work is to preprocess flight strip information and construct a symbolic representation of potential collisions and an aircraft neighborhood of each collision pair. The algorithm used is based on development work accomplished at the Mitre Corp. (Kingsbury [35]) and made more efficient my the inclusion of several heuristics used by controllers (Chien et al [17]). The conflict filter equations are described in Appendix B. The approach is to create a space of all possible airplane pairs and search the space for pairs that violate a user specified horizontal and vertical separation standard during a user specified time window. The heuristics filter airplane pairs from the space. For instance, one heuristic indicates that aircraft on routes that are known not to intersect do not violate the separation standard. The nonintersecting airway knowledge is contained in the map data base.

There are four horizontal separation standards which correspond to the possible collision types: head-on, crossing, merging, and overtake. A numerical value for each standard is user specified, and I have used the values 10, 10, 10, and 20 nautical miles, respectively. Controllers were required to keep aircraft 5 miles apart before the controller's strike in the summer of 1981. Immediately after the strike, the standard



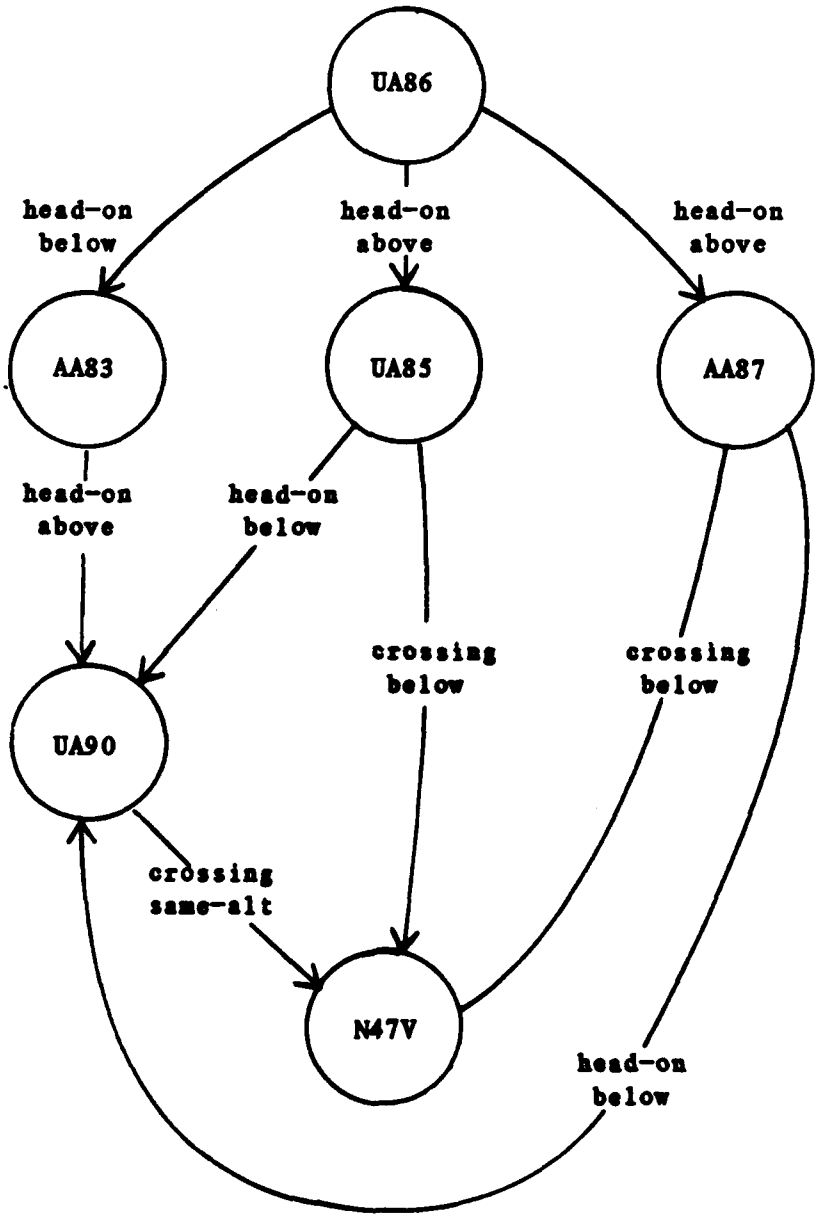
Radar Display of a Typical Aircraft Conflict Scenario
Figure 5

changed to 20 miles and now is 10 miles. The vertical standard is 2000 feet for aircraft above a mean sea level altitude of 29,000 feet and 1000 feet for other aircraft. The preprocessor output for Fig. 5 (p. 33) is shown below. The syntax of each conflict entry is a list that specifies the conflict time, the conflict type, aircraft-1, aircraft-1's relative altitude to aircraft-2, aircraft-2, and the relative miss distance (nautical miles).

```
(setq *conflicts*
 '((18.596 head-on   ua86 below     aa83 5.92E-06)
   (18.644 head-on   ua86 above     ua85 0.64)
   (18.655 head-on   aa83 above     ua90 5.44E-06)
   (18.677 head-on   ua86 above     aa87 1.42)
   (18.693 crossing ua85 below     n47v 7.15)
   (18.704 head-on   ua90 above     ua85 2.38)
   (18.715 crossing ua90 same-alt n47v 9.37)
   (18.727 crossing aa87 below     n47v 3.56)
   (18.735 head-on   ua90 above     aa87 1.96)))
```

The output of the conflict filter is ordered based on the aircraft with the most conflicts and then discriminated. The assumption is that a 'real' conflict filter would be operating in parallel with the expert system and able to detect any conflict situation far in advance. The strategy implemented here concentrates on a more subtle aspect: recognizing the relationship between conflict pairs. The resulting discrimination network is referred to as a conflict structure.

The conflict structure for the previous example is shown in Fig. 6 (p. 35). The nodes represent aircraft and the links represent the type of conflicts between them. Additional contextual information such as the aircraft intent and relative altitude are shown. Unidirectional links are specified, but the attached attributes can be inverted. For instance, the fact that ua86 is in a head-on conflict with and below



Conflict Structure
Figure 6

aa83 implies that aa83 is in a head-on conflict with and above ua86. The conflict filter is also used to find a neighborhood of non-conflict aircraft about each conflict pair. Symbolic horizontal and vertical descriptions (e.g., left-of, above) are used to describe the neighborhood of each conflict aircraft. Again, unidirectional linkages are specified and the attributes are invertible. The neighborhood data are stored on the property list of each aircraft. The data are useful during the problem solving phase for command selection and critique.

2.0. Problem Recognition and Decomposition

Each link of the conflict structure specifies an aircraft conflict goal of the form:

```
(or () (vertical-sep   ?ac1 ?ac2)
       *vertical-separation*)
    () (horizontal-sep ?ac1 ?ac2)
       *horizontal-separation*))
```

The conflict structure can be decomposed into a into a set of two airplane problems. Examples of this approach in the air traffic control domain are Wesson's production system [65] and an algorithmic planning system (Rucker [53]). The expert system defaults to this behavior in the absence of higher domain concepts. For example, consider the problem illustrated in Fig. 5 (p. 33) would require the solution of nine subproblems and the search through the resultant solution space. I have seen this behavior in novice controllers. The experienced controller has acquired concepts at a higher level and can solve problems involving many aircraft. The example problem was taken from a training exercise currently used at the Chicago ARTCC. Significantly, a major objective

of the exercise is to teach novice controllers higher level problem-solving concepts. Experienced controllers I interviewed perceive this problem as three independent problems. Fig. 5 (p. 33) will be used for illustration throughout the remaining portions of this chapter.

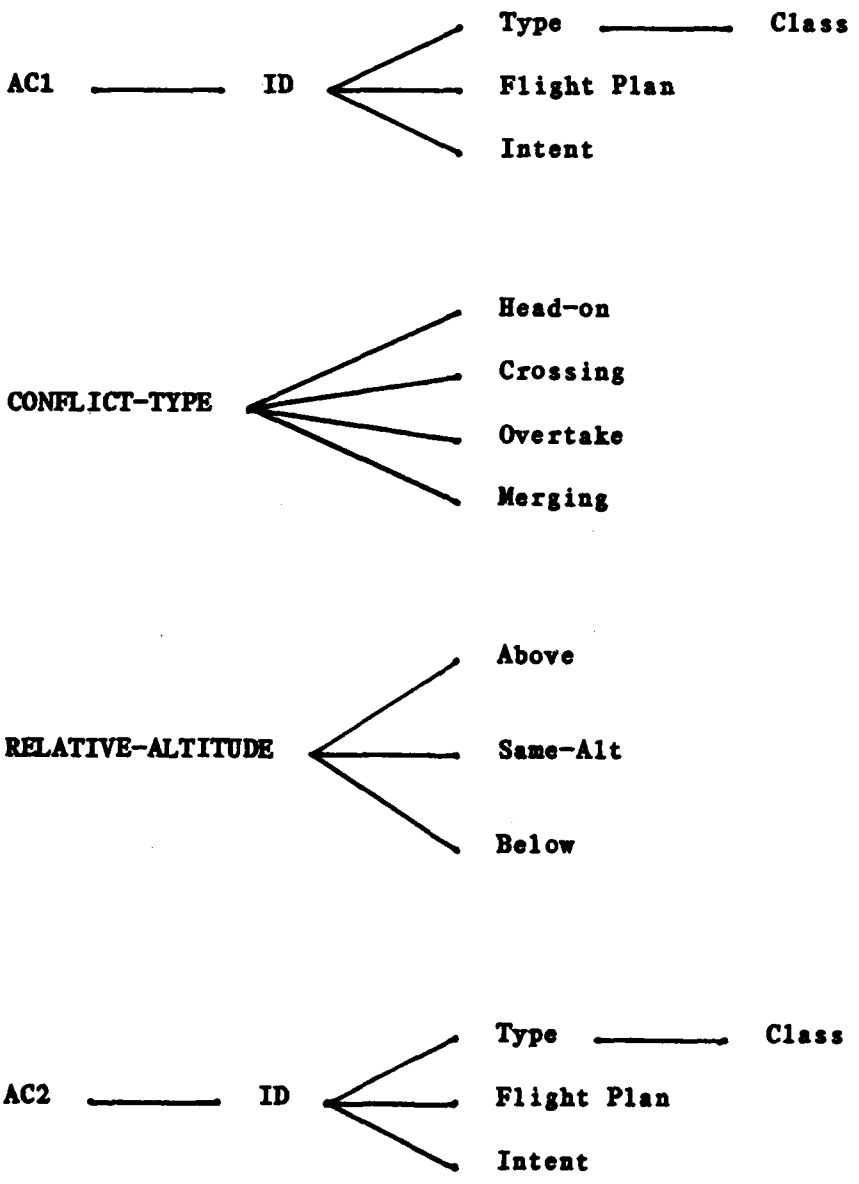
Many types of goal interactions are evident in enroute air traffic control, some of which are explicit in the conflict structure. The conflict structure syntactically defines recurring goals in the sense defined by Wilensky [66]. Recurring goals involve a common subject in repeated and similar conflict situations. For instance, aircraft ua86 is involved in similar conflicts with aircrafts aa83, ua85, and aa87. Another type of goal interaction is goal overlapping. Goal overlapping occurs when a goal (or goals) involving one subject is similar to a goal (or goals) of another subject. Note that both aircraft ua86 and ua90 have identical collision avoidance goals: to avoid conflicts with aircraft aa83, ua85, and aa87.

The goal conflict problem is not transparent in the conflict structure. Goal conflicts involve situations where plans to resolve one goal create or prevent the achievement of other goals. There are several instances in air traffic control. First, the plan to achieve a collision avoidance goal may create additional conflicts. These situations can be detected by critics that apply the conflict filter to proposed plans. The neighborhood of each conflict pair improves the efficiency of this search since only the aircraft in the neighborhood need be examined.

A more subtle aspect is that aircraft usually have the goal to remain fuel efficient. While not always true for military aircraft or for aircraft that have declared emergencies, it is generally an active

goal for commercial aircraft in controlled enroute airspace. The human controller has a strong bias towards safety and when the safety and expediency goals conflict, he tries to formulate a plan that minimally degrades the expediency goals while achieving the safety goals. The knowledge is implicit in his problem solving strategies which will now be discussed.

The approach to problem decomposition follows the 'linear assumption' of HACKER (Sussman [62]), NOAH (Sacerdoti [54]), and other early planning systems. The approach also seems consistent with human controller planning behavior. Simon suggests this is a fundamental approach to all human problem solving [57]. The decomposition strategy consists of focusing the expert's attention on key structural attributes of the conflict network and then filtering those attributes to achieve a semantic interpretation of the structure. The focus of attention mechanism relies on syntactic pattern matching. The templates for the two-aircraft and recurring-conflict problems are shown in Fig. 7 (p. 39). Note that there are several instances of recurring conflicts in the example. ua86 and ua90 are involved in head-on conflicts with aa83, aa87, and ua85. The filtering phase consists of understanding how the specific attributes of the substructure affect the problem solving behavior. Each template matches a strategy and instantiation of that strategy establishes the context within which further problem solving will be accomplished. I say that the computer understands the problem in the same way that language comprehension researchers say that application of an appropriate script demonstrates understanding. If the global structural description cannot be comprehended, it is decomposed and each resultant



Strategy Matching Templates
Figure 7

subprocess is again input to the decomposition process.

3.0. Problem Resolution

The problem resolution phase consists of inferring goals and plans to achieve those goals given a structural description. Each template has an associated strategy which is invoked at this point. There may be more than one strategy associated with a given pattern matching template. The strategy represents a semantic interpretation of the particular substructure and contains the knowledge required to direct problem solving. The strategy also contains the knowledge necessary to decide if it is applicable in this context. For instance, there are two problem solving strategies for the case where there are two aircraft in a head-on conflict. One strategy will choose one of the aircraft to be subject to a command. The other strategy recognizes situations where there are uncongested neighborhoods and issues commands to both aircraft. Invoking a strategy is equivalent to understanding the substructure. If the expert system is used to control aircraft, the resultant plan is offered for execution. If the expert system is asked why a controller might control aircraft in this manner, the resultant plan is the explanation.

The knowledge of the recurring conflicts concept was used to decompose the problem into five (as opposed to nine) subproblems as shown in Fig. 8 (p. 41). Strategy h00022 is an instance of a head-on recurring strategy. It directs that ua86 be subject to resolution commands and that a common command be given to resolve the conflicts with ua85 and aa87. The preferred command sequence is turn, climb, descent, and holding pattern. A turn is rejected (actually the turn tactic did not exist when this problem was examined). Similarly, a climb command is chosen

```
conflict structure decomposed into 5 subproblems
************************************************

concept      instance          conflict   rel          involved
 name          of               type     alt          aircraft

h00022  recurring-conflicts   head-on    above     ua86 with (ua85 aa87)
h00021  recurring-conflicts   head-on    above     aa83 with (ua86 ua90)
c00020  recurring-conflicts   crossing   above     n47v with (ua85 aa87)
h00019  recurring-conflicts   head-on    above     ua90 with (ua85 aa87)
c00018  crossing-two-aircraft crossing   same-alt  ua90 with (n47v)

;
;  finding a resolution command
;
-> (rc h00019)

creating a new strategy using recurring-conflicts concept
p00023 is a modified instance of head-on-two-aircraft
;
;  fuel efficient commands are specified
;
fuel efficient command preferences:

  (ua90 turn ua85 aa87)
  (ua90 climb ua85 aa87)
  (ua90 descent ua85 aa87)
;
;  invoking operators
;
t00024 is an instance of a turn operator
turn does not pass tactical criticism

c00025 is an instance of a climb operator
climb does not pass tactical criticism

d00026 is an instance of a descent operator
desired subject altitude = 37000.0
applying descent-rule-23
(maintain 37000.0 until 18.70455365521318)
tactical critics: conflict-filter descent-performance
descent passes tactical criticism
```

Solution Using Recurring Patterns
Figure 8

since jet aircraft are usually more fuel efficient at higher altitudes. An appropriate climb command is given to ua86.

4.0. Plan Assembly and Critique

Commands are tested by procedural critics at the tactical and the strategic levels. At the tactical level, the performance capabilities of the aircraft are examined with regard to the proposed command. There is a limited capability to modify the command if the performance is not satisfactory. For instance, the time a command is to be issued can be modified, if an aircraft's climb rate would be excessive. At the strategy level, the expert insures that the proposed command does not cause additional conflicts with aircraft in the subject aircraft's neighborhood. The strategy critic may make suggestions to its instantiated tactic. For instance, it may suggest that the command time be modified.

5.0. Knowledge Representation

I have chosen to represent the controller's knowledge in frames. Intelligent processing requires both large and small chunks of knowledge in which individual molecules have their own substructure. Minsky's paper [44] on frames provides the theoretical foundation. Others have implemented frame languages (Bobrow [7], Bobrow and Winograd [8], Roberts and Goldstein [52], Charniak et al [15], and Stefik [59]). The fact that I choose to use frames or write a frame language is not a novel idea, except for the possibility of showing how to integrate diverse knowledge in a novel task domain. Frames are used to represent collections of information at many levels within the system. Some frames describe the detailed knowledge required to implement resolution com-

mands, while other frames describe high-level conceptual knowledge. A frame is a data structure that contains a name, a reference to a prototype frame, and a set of slots. Frame names are primarily mnemonic devices and are not used in the reasoning process.

A frame may be a parent (called a prototype) and a child (called an instance). A frame's important substructures and its relation to other frames is defined in its slots. A slot has a slot name, a filler or value, and possibly a set of attached procedures. The value of a slot may be another frame or in the case of a prototype, a description constraining what may fill the corresponding slots in any instance of the given frame.

Procedures are attached to a slot to indicate how certain operations are to be performed which involve the given frame or the corresponding slot in its instances. The procedures associated with slots fall into two general classes: servants and demons. Servants are procedures that are activated only on request. Demons are procedures that are activated automatically when a datum is inserted into an instance. The air traffic control knowledge is classified as knowledge about aircraft, knowledge about tactics, and knowledge about strategies.

The frame language designed for the expert system uses twelve types of aspects which are now briefly described.

1. Equal aspect (=). An equal aspect indicates the value of a slot and is represented as the dotted pair '(= . value).' The significance is that the equal aspect of a slot can be overridden by a value in a lower level frame.

2. Always aspect. An always aspect is similar in purpose to an equal aspect, except that always aspects cannot be overridden.

3. Must-be aspect. A predicate function that specifies the values that a equal or always slot may have. For instance, the value of an aircraft's velocity must be greater than 250 knots which is an FAA constraint in enroute airspace. Attempts to insert values into an equal or always aspect of a slot that violate the must-be constraint fail.

4. Must-not-be aspect. This is similar to the must-be aspect except that it contains the negation of the predicate function that might otherwise be specified in the must-be aspect.

5. Should-be aspect. A predicate function that functions exactly like a must-be slot, but allows the addition of equal or always slot values. The primary usage is to print out warning messages.

6. Should-not-be aspect. The should-not-be aspect is analogous to the must-not-be aspect.

7. If-needed aspect. The slot is also called a servant. The if-needed aspect specifies a procedure that can be evaluated when a slot is queried for an equal or always aspect, but evaluates to nil. For instance, a primary usage in the tactic frames is to specify the rules which should be searched when a resolution command is required.

8. If-added aspect. The if-added aspect is also referred to as a demon. It is the primary repository for critics that are evaluated when a value is attempted to be inserted into another aspect. Its purpose is similar in function to the predicate procedures defined in must-be or should-be aspects, but the if-added aspect procedure is more global. It

can not only be used as a critic, but can be used to create aspects in different frames. In this language, it is referenced only when a new slot is defined.

9. If-modified aspect. The if-modified aspect procedure is similar in function to the if-added aspect, except that it effects existing slots.

10. If-deleted aspect. The if-deleted aspect procedure is similar in function to the if-added procedure, except that it becomes active only when slot values are deleted.

11. If-accepted aspect. The if-accepted aspect defines message handling procedures that handle communication between different knowledge levels and between frames. For instance, if a tactically proposed plan passes tactical criticism, it is scheduled for processing by its strategy's critic.

12. If-rejected aspect. This aspect is similar to the if-accepted aspect. It specifies the processing required if a critic finds fault in a plan.

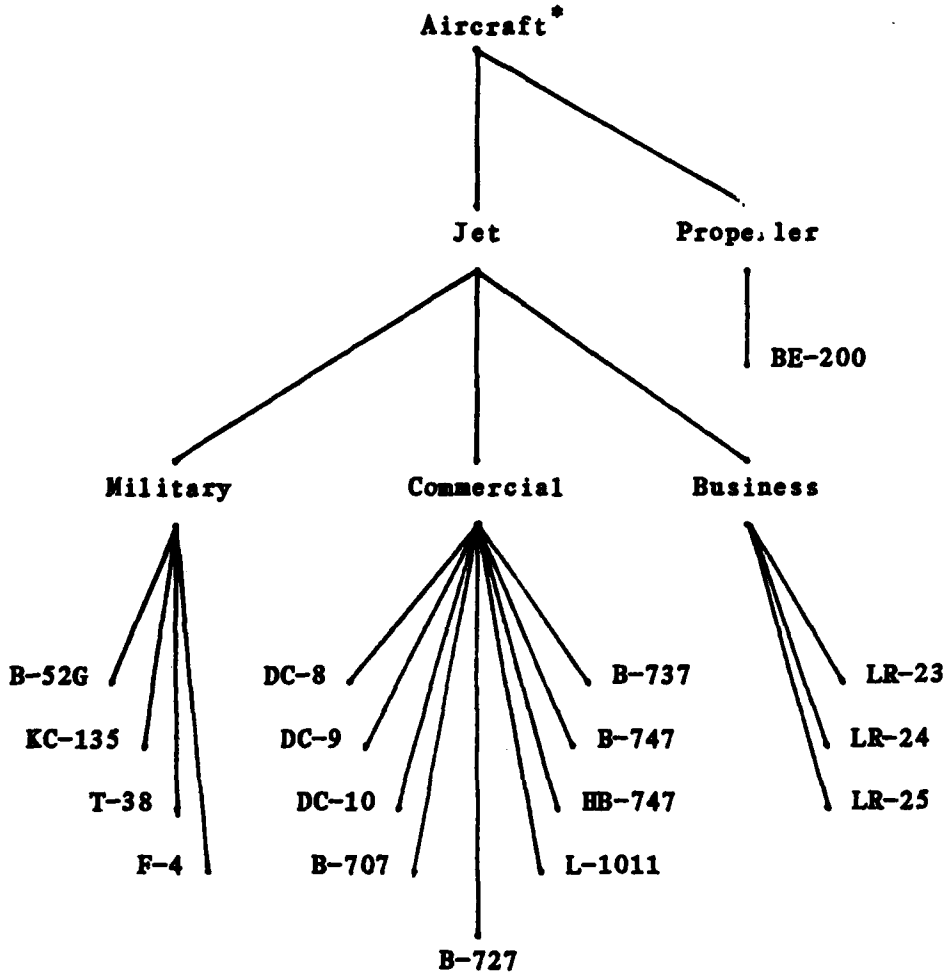
5.1. Aircraft Knowledge

A controller has a qualitative understanding of aircraft capabilities. For instance, he knows that jet aircraft generally are more fuel efficient at higher altitudes. This type of knowledge is represented in an aircraft frame. In this instance, frames provide an interface between mathematical knowledge and heuristic knowledge. For instance, the knowledge that jet aircraft generally like to fly at higher altitudes is represented by an equation called 'optimum altitude' which has the vari-

ables weight, equivalent surface area, air density, and wing size. The equation is stored in a servant slot in the jet aircraft frame. A particular jet aircraft, say a Boeing 747 (B747), is an instance of a jet. The frame for a B747 will contain knowledge that is relevant to that aircraft (e.g., frontal surface area, wing length, weight, etc.). A controller's concept of an aircraft also includes knowledge about possible changes in aircraft variables. For instance, the controller knows that wing size is constant and that the angle of attack is controlled by the pilot. A particular instance of a B747 would be an aircraft that is presently flying in the control sector. The hierarchy of aircraft frames is shown in Fig. 9 (p. 47). Presenly the aircraft knowledge base consists of 23 aircraft. A representative frame for a B747 is shown in Fig. 10 (p. 48).

5.2. Tactical Knowledge

I have represented the tactical-level knowledge in five tactical frames. Each contains the knowledge required to implement an operator to satisfy a specified goal. For the purposes of discussion, consider the climb prototype frame of Fig. 11 (p. 49). When an instance of this frame is created, the conflict aircraft and conflict context (time, place, type) are defined. The goal then is to find an appropriate climb command so that the 'subject-to-command' aircraft will no longer conflict with the 'right-of-way' aircraft. There are 'common sense' constraints which would immediately rule out the the climb command (e.g., the 'subject' is descending beneath the other aircraft). An appropriate command is chosen based on the subjects proximity to navaids, present state vector, and capabilities. The command is criticized to insure that aircraft



*References a knowledge base of engine types
(reciprocal, turbojet, turbofan, turboprop)

Hierarchy of Aircraft Frames
Figure 9

```
(frame jet-aircraft an aircraft with
       (fuel-flow-rate always "proportional to thrust required"))

(frame commercial-jet-aircraft a jet-aircraft with
       (parasitic-drag-coef = 0.012)
       (efficiency-factor =  0.8))

(frame b-747a a commercial-jet-aircraft with
       (chord = 196.0)
       (aspect-ratio = 6.96)
       (wing-area = 5500.0)
       (weight-empty = 360000.0)
       (weight-takeoff = 520000.0)
       (engine-type = JT9D-7A)
       (engine-number = 4))

(frame hb-747a a b-747a with
       (descriptor = heavy)
       (weight-empty = 442000.0)
       (weight-takeoff = 775000.0))
```

Frame Representation of a B-747
Figure 10

```
(frame tactic a knowledge-representation with

    (command-status if-rejected

        (add-slot *strategy* 'command-status '= 'rejected))

    (command-status if-accepted

        (progn
               (add-slot *strategy* 'resolution-command '=
                         (slot-val *tactic* 'resolution-command))
               (add-slot *strategy* 'command-status '= 'accepted))))


(frame climb a tactic with

   (cmd-issuance-time   if-needed    (climb-cmd-issuance-time))

   (desired-altitude    if-needed    (desired-climb-altitude))

   (opt-climb-rate      if-needed    (opt-climb-rate))

   (max-climb-rate      if-needed    (max-climb-rate))

   (climb-rate          if-added     (climb-rate-constraint))
   (climb-rate          if-modified  (climb-rate-constraint))

   (resolution-command        if-needed    (find-climb-command))

   (proposed-commands       if-added     (climb-critic)))
```

Tactic Frames
Figure 11

performance capabilities are not exceeded and the command resolves the conflict and does not cause any more. There are 17 different commands that could be given to implement a climb depending on an aircraft's present state vector, its proximity to navaids, and the flight path of the conflicting aircraft. Commands are represented as rules and an if-needed procedure specifies the rules to use for each tactic. Other tactics contain special algorithms. For example, the turn tactic utilizes an algorithm called the 'wedge of prohibited angles' to compute the minimum turn angle when a subject aircraft is to be vectored around another (Kingsbury [35]).

5.3. Strategy Knowledge

The conflict resolution strategies are based on problem-solving strategies that novice controllers are taught for two-aircraft problems. Given a two-aircraft problem, the expert designates an aircraft that will be subject to resolution commands and chooses a preferred and default sequence of commands that form a skeleton plan. This knowledge is in the form of rules and tables indexed on the conflict type, the aircraft intents and the relative aircraft state vector. The rules used to determine the subject and right-of-way aircraft are listed below.

1. If the involved aircraft have the same intent, then the subject aircraft is the aircraft farthest from the conflict point.

2. If the involved aircraft have the same intent, then the subject aircraft is the aircraft at the lower altitude.

3. An aircraft in level flight has the right-of-way over an aircraft changing altitude.

4. A climbing aircraft has the right-of-way over a descending aircraft.

5. An enroute-descending aircraft has the right-of-way over an arrival aircraft.

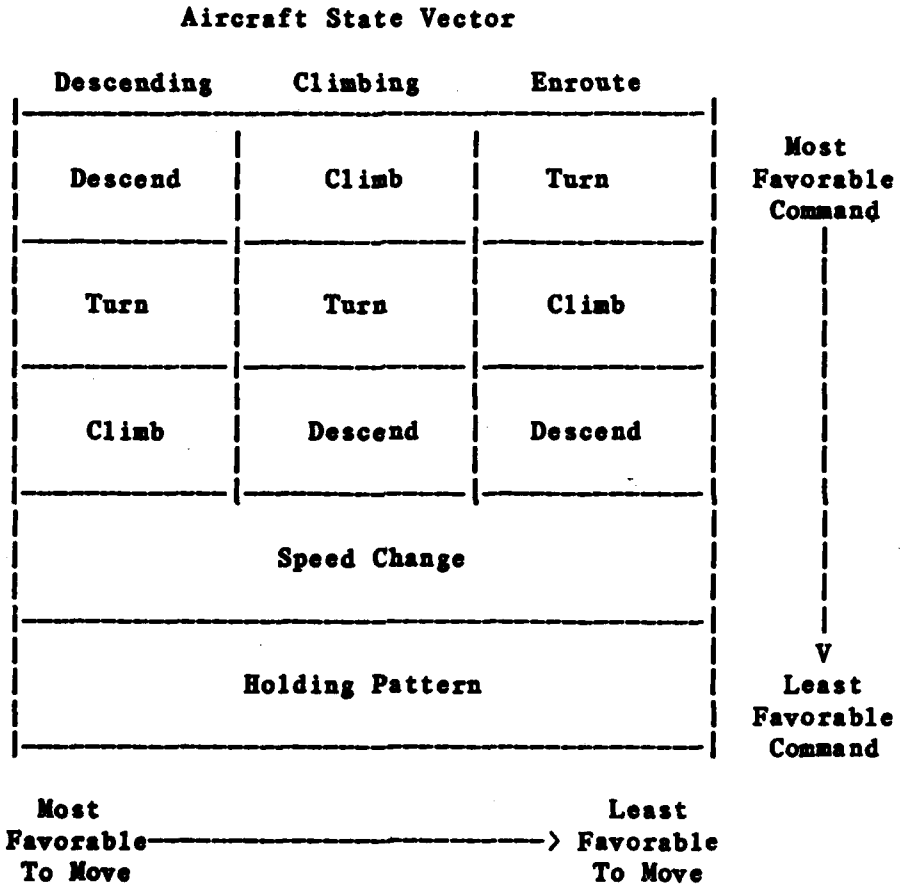
6. An enroute-climbing aircraft has the right of way over a departure aircraft.

7. If no other rules apply, designate one aircraft as the subject and the other as the right-of-way.

Fig. 12 (p. 52) illustrates the command preferences for a crossing conflict. An example strategy is shown in Fig.13 (p. 53) for a head-on conflict. The 'when-active' slot specifies the context in which this strategy should be activated. Implicit in these frames is the knowledge that safety always is more important than fuel efficiency. The preferred commands prioritize their commands on the basis of fuel efficiency.

6.0. Discussion

A complete solution for the problem illustrated in Fig. 4 (p. 29) is given in Appendix C. In this chapter, I have described the strategies for defining, decomposing, resolving, and critiquing conflict avoidance problems. I have described the use of frames for the representation of a controller's heuristic knowledge about problem solving strategies, tactics, and general knowledge about aircraft. The original thesis claimed that the computer required the ability to reason about its plans, to justify its responses, and to acquire new knowledge. In the domain of enroute air traffic control, these claims parallel the measures of problem solving elegance that controllers use to describe 'good air traffic control practice.' In summary, those measures involved using, recognizing, and understanding plans that localized conflicts, relaxed constraints, and handled multiple goals. It now seems pertinent to ask how close does the expert system come to achieving elegance and how much closer would a plan justification capability bring it?



Crossing Conflict Command Preferences
Figure 12

```
(frame resolution-strategy a knowledge-structure

       with

       (instances = (recurring-conflict-strategy two-aircraft-strategy))

       (resolution-command if-needed  (resolution-command *frame*))

       (command-status if-rejected (reject-command))

       (command-status if-accepted (accept-command))

       (cmd-issuance-time if-needed (command-issuance-time))

       (prohibited-altitudes  if-needed  (prohibited-altitudes))
       (prohibited-headings   if-needed  (prohibited-headings))
       (prohibited-velocities if-needed  (prohibited-velocities))

       (minimum-altitude = 24000.0)
       (maximum-altitude = 45000.0)
       (maximum-heading-change = 45.0))

(frame recurring-conflict-strategy a resolution-strategy with

    (instances = (head-on-recurring-strategy crossing-recurring-strategy
                  merging-recurring-strategy overtake-recurring-strategy))

    (when-active = (and (listp cnflct-type)
                        (listp row-ac)
                        (> (length row-ac) 1))))

(frame head-on-recurring-strategy a recurring-conflict-strategy with

    (when-active = (and (eval (slot-val 'recurring-conflict-strategy
                                        'when-active))
                        (> (num-conflicts-of-type '(head-on)) 1)))

    (preferred-commands if-needed (head-on-preferences
                                       (slot-val *strategy* 'sbjct)
                                       (slot-val *strategy* 'tcnflct)
                                       (slot-val *strategy* 'row))))
```

Strategy Frames
Figure 13

I defined conflict localization as an intangible human skill and implemented algorithms to preprocess flight data. The data were used to construct a symbolic representation of the potential collision. Many goal interactions could be syntactically found by reference to the diagram. This is analogous to the controller's ability to rapidly digest a traffic scenario and visualize the conflicts. The problem solving strategies were developed after interviewing air traffic controllers, visiting controller training facilities, and reviewing training material. Additionally, the knowledge representation chosen integrates the various types of heuristic knowledge the controller uses. The expert system can be improved by implementing a plan justification capability that allows the computer to reason about its plans.

CHAPTER V

AN APPROACH TO QUALITATIVE REASONING

Qualitative reasoning is the process of drawing conclusions and inferences from possibly incomplete observations, data, and knowledge. For instance, one knows that a large aircraft requires a greater propulsive force than a small aircraft if both are to maintain the same velocity. If one wishes to determine how much more force is required, one needs to be given the masses of the respective aircraft, the desired velocity, and the applicable aircraft equations. de Kleer [25] explored the qualitative/quantitative knowledge dichotomy in NEWTON, an expert 'roller coaster' problem solver. He created an envisioning process based on qualitative knowledge that completely described future system states and directed the application of quantitative knowledge. Recent research (de Kleer [27], Forbus and Stevens [32], Forbus [31], Kuipers [37]) has sought to increase the qualitative reasoning capabilities by including knowledge about the behavior of system variables. For instance, Forbus represents each quantity in terms of the sign and magnitude of both its amount and derivative. In this chapter, I will develop a qualitative reasoning capability that allows the computer to reason about aircraft performance goals.

The expert system described in the previous chapter requires a qualitative reasoning capability for two reasons. First, a controller's plans should be justified with well-founded reasons. Many times the reasons why particular plans are used are only understood by the controller qualitatively. For instance, a controller understands an aircraft is

more fuel efficient at higher altitudes. He may not be able to provide a proof of this heuristic but would probably understand an explanation that included the facts that the resistive force encountered by the aircraft was decreased because air density decreases as altitude increases. This illustrates another reason for the qualitative reasoning capability. As new equations and algorithms are developed, there is a need for the computer to be able to understand their intended use and be able to generate explanations that are comprehensible to the human controller.

The approach developed here is based on a qualitative understanding of Newton's laws applied to one-dimensional translational motion. The approach differs from previous approaches in three aspects. First, it is a bidirectional process. While previous approaches have emphasized a top down approach from qualitative knowledge to quantitative knowledge, I implemented a capability that allows reasoning through a common knowledge base. The common knowledge base is a 'naive physics' representation of Newton's laws and associated primitive concepts (e.g., force, motion, mass, power). The reasoning capability allows the expert system to justify its plans. That is, the computer can construct well-founded explanations for its planning actions based on the interpretation of detailed domain equations. Second, the naive physics knowledge adds a level of abstraction to previous techniques. The motivation is that most people (certainly controllers) understand Newton's laws but may have difficulty comprehending domain dependent equations and algorithms upon which they are based. As in de Kleer's early work [25], the qualitative reasoning capability allows the computer to find the pertinent equations that should be used when quantitative reasoning is required. Forbus

utilizes knowledge about the sign and magnitude of a quantity's amount and derivative. The approach here is similar, with the added feature that a quantity can vary as a function of many variables. The qualitative reasoning process supports a limited form of sensitivity analysis. By limited, I mean that the sensitivity analysis is restricted to one dimension. However, any number of variables can be varied and their influence evaluated. A brief example from air traffic control will be illustrative.

Assume that a controller's collision avoidance plan requires an aircraft to climb. Is the plan fuel efficient? A quantitative approach would require computation of fuel used with and without the plan. This is a sufficient procedure, but qualitative knowledge can be used to simplify some problems and provide an 'insight' that might not otherwise be possible. From a qualitative viewpoint the plan is fuel efficient if less propulsive force is required. A jet aircraft's propulsive force (or thrust) is proportional to the fuel flow rate (a constant called specific fuel consumption is calibrated for each engine). A qualitative comparison can be made based on the knowledge of aircraft states and state changes. For example, an aircraft that climbs from a level state requires an initially larger thrust, but if the altitude at which it levels off requires less thrust and if the aircraft intends to stay at that altitude for a long time, then it can be concluded that the plan is fuel efficient. When ambiguities are encountered, the naive physics knowledge can be used. Approximate computations can be made for how long an aircraft is in transition and assigned a qualitative value. For instance, a climb that takes approximately one minute may be called 'a

short time.' The interface with the conceptual level knowledge base will define what is meant by the notion of 'a short time' for a particular domain.

1.0. Naive Physics Knowledge

Dynamic systems such as aircraft and automobiles are either at an equilibrium condition or in the process of moving to a new equilibrium condition. An equilibrium condition satisfies Newton's second law. That is, when a system's applied forces counterbalance inertia ($F = ma$), the system is in dynamic equilibrium. Equilibrium conditions are called states and intentional perturbations from states are called state changes. For instance, an aircraft in level, unaccelerated flight is said to have the 'level state.' I restrict attention to intentional state changes where intentional state changes form the detailed segments of plans. Plans are intentional actions to achieve goals. The envisioning process means that the computer can decompose a plan into a series of states and state changes, and apply qualitative knowledge to resolve some ambiguity and direct the application of pertinent quantitative knowledge.

Newton's laws are represented in a semantic network where nodes are primitive concepts (e.g., force, velocity) and links indicate their interdependence. The representation is motivated by Rieger [51] but is purposively less ambitious for two reasons. First, I only wish to represent my understanding of Newton's laws as they pertain to the one-dimensional motion. Second, the representation of the naive physics knowledge only provides the illusion of automated understanding (much in the same sense as explanations formed by the recitation of invoked

production rules). The naive physics knowledge is made explicit by the designer. The linkages between the naive physics level and a domain are equation dependent. The representation serves as the basis for the interpretation of yet-to-be specified equations and algorithms. Thus, automated understanding is achieved by the ability to transfer and translate knowledge between two diverse knowledge bases (one conceptual and one mathematical).

The representation is shown in Fig. 1 (p. 8). Nodes represent concepts in physics. I have represented nine concepts: force (propulsive, enabling, resistive), motion (acceleration, velocity, position), mass rate of change, mass, and power. Propulsive and enabling forces are referred to as positive forces. Motion variables are said to be qualitatively proportional to positive forces since the sign of the change of motion variables varies in accordance with the change in sign of positive force variables. Propulsive forces require an external enablement which converts fuel (portion of system's mass) into the force. The rate of change of mass varies indirectly with changes in propulsive force. That is when propulsive force increases, the fuel flow increases which causes the total mass rate of change to increase. Another positive force is called an enabling force. Enabling forces do not directly influence mass. For instance, the air flow over a wing causes a pressure differential that enables lift. Lift is an enabling force that counteracts weight. Resistive forces counteract positive forces. Common examples are pressure and friction forces which vary with velocity, weight, density, and domain dependent variables such as surface size. The motion variables are related by a weak notion of integration that simply states

when acceleration encounters a positive change so do velocity and position. When mass decreases, acceleration tends to increase. As will be described this is referred to as a secondary affect. Power is represented as a function of propulsive force and velocity. While the physical concept of power is related to the work done by a system's composite forces, I have represented only the influences due to propulsive force. Each node is represented by a frame. Several of the nodes in the naive physics representation are shown below.

```
(frame Fprop a node w
    (name = propulsive-force)
    (influences = (F+))
    (level = naive-physics)
    (instance = ((horz-level thrust))))

(frame Fres a node w
    (name = resistive-force)
    (instance = ((horz-level drag) (vert-level weight)))
    (level = naive-physics)
    (influences = (F)))

(frame vel a node w
    (name = velocity)
    (influenced-by = ((primary+ accel)))
    (level = naive-physics)
    (influences = (pos))
    (instance = ((horz-level vh))))
```

There are four types of influence links: primary and secondary positive influence and primary and secondary negative influence. For example, acceleration is primarily influenced by force and secondarily influenced by mass. Note that this is my interpretation of the physics. Acceleration usually changes in a dynamic system because one changes the applied force, although there are instances when the mass is changed (e.g., discarding ballast in hot air balloons). Additional concepts can

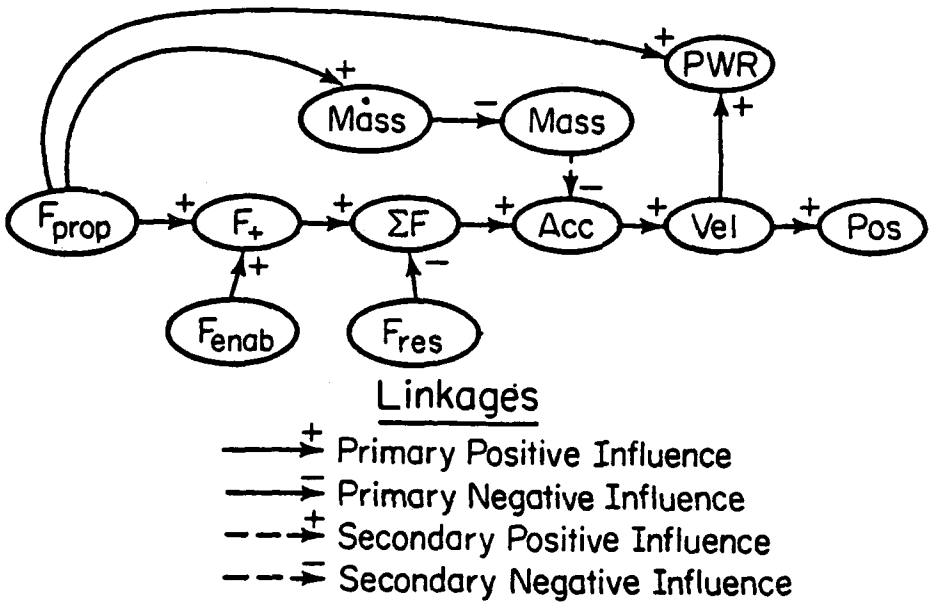
be added. For instance, specific examples of resistive force are pressure force and friction. The concept of pressure force would indicate that its influence is velocity. Additional nodes define concepts like equilibrium and show when to initialize demons. This knowledge is attached to the slots of frames. In particular, when the conditions for equilibrium are violated, position and time demons are activated. These demons compute the time and distance involved in achieving a new equilibrium condition. The computations are approximations of the 'real world' because the exact equations are not used. A demon mechanism is used to compute the affect on nodes when perturbations from equilibrium are encountered. The approach is motivated by linear system theory (Chen [16]).

1.1. Propagating Qualitative Values

The diagram in Fig. 14 (p. 62) represents the designer's abstract understanding of Newtonian mechanics. In a manner reminiscent of de Kleer's propagation of Incremental Quality (IQ) values in electronic circuits (de Kleer [26]), three qualitative values are designated: increase (incr), decrease (decr), and no-change (nc).[2] Many queries can be handled by propagating these qualitative values in the network. For instance, one knows that if one wishes to increase velocity, then one can increase the positive forces, decrease the resistive forces, or decrease the mass. A trace is shown below.

---

[2] I have not assigned the value 'ambiguous because, if required, the value can be computed from the domain equations.



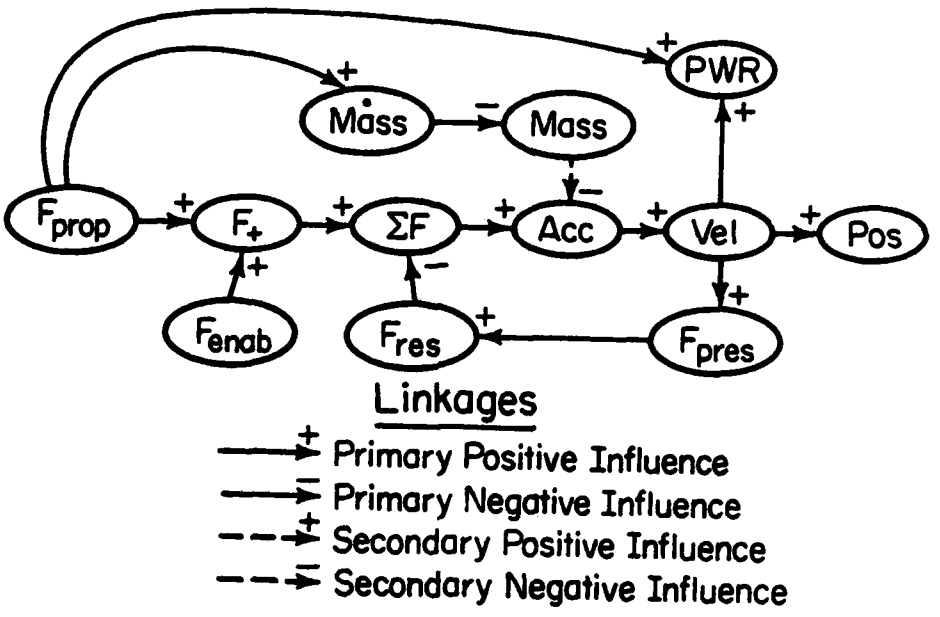
A Naive Physics Representation of Newton's Laws
Figure 13

```
   ; define goal to have velocity increase
-> (perturb 'vel 'incr)
   incr

   ; what would cause the increase?
-> (backtrace-through-nodes 'vel)
   vel

   ; a list of plausible influences
-> *possible-influences*
   ((incr Fprop Fenab) (decr Fres))
```

An example involving feedback is illustrated in Fig. 15 (p. 63). Pressure force is qualitatively proportional to velocity. Feedback is handled qualitatively, using the idea of 'clash' and 'coincidence' as



Pressure Force Illustrates Feedback
Figure 15

proposed in the STEAMER project (Forbus and Stevens, [32]). A clash (negative feedback) occurs if some rule tries to set a quantity to a value different than a value obtained by another means. A coincidence (positive feedback) occurs if some rule tries to set a quantity to a value equal to that obtained by a different means. Suppose the positive force increases. This causes a corresponding increase in the motion variables. An increase in velocity causes pressure force to increase which in turn causes the resistive force to increase. Since resistive force is a negative influence on total force, a clash is encountered.

Clashes and coincidences are easily detected by counting the number of 'influence inversions' when a loop is traversed. For instance, the feedback loop has one inversion. Odd numbers of inversions indicate negative feedback, while even number of inversions indicate positive feedback. In some cases, it is sufficient to find evidence of negative feedback. In other instances, more detailed knowledge is required.

1.2. Ambiguity Resolution Using Approximate Computations

How long does it take for the system to reach a new equilibrium value? What are the equilibrium values? These types of questions require the use of quantitative knowledge. Approximate computations using tangential models or linearized equations can be efficient and are consistent with previous qualitative reasoning approaches. The diagram of the naive physics can be treated as a linear control system and perturbation analysis techniques can be applied. The solutions are represented explicitly at the naive physics level. Consider the feedback loop in Fig. 15 (p. 63). The equation for a pressure force is shown below.

$$F_{pressure} = K V^2 \qquad (5.1)$$

where V is velocity and K represents domain dependent terms that contribute to the pressure force. In the aircraft domain, these terms include air density and the surface area that is normal to the velocity vector. When the system is at equilibrium, the pressure force equals the positive force. A change in the positive force causes a perturbation in the pressure force. The transient has the form of an exponential decay with a time constant equal to:

$$\text{time constant} = \frac{\text{mass}}{2 K V_o} \tag{5.2}$$

where the pressure force has been linearized about the former operating point: V_o. Then an expression for velocity is:

$$V(t) = V_o + \frac{F_{pos}}{m} \left(1 - e^{\frac{-2KV_o t}{m}}\right) t \tag{5.3}$$

The knowledge of transients is useful for understanding the time-elapsed behavior of aircraft performance. From the time constant, it is apparent that heavier aircraft require longer to change their velocity.

1.3. Consistency with Newtonian Mechanics

The abstract representation of Newton's laws presented in this section is claimed to be consistent with Newton's laws. Consider the case where there are no applied forces. Then qualitatively, it can be stated that velocity does not change (Newton's First Law). Suppose that a force is applied. Then the resultant acceleration varies proportionally in the same direction (Newton's Second Law). The applied forces interact with something which apply an equal and opposite force. For instance, an applied positive force may create an equal and opposite resistive force (Newton's Third Law).

2.0. Qualitative Knowledge About Aircraft Performance

Subsonic aircraft performance capabilities can be computed from the drag polar shown in Fig. 4 (p. 29) and knowledge about engine performance. For example, an aircraft's maximum velocity occurs when its max-

imum positive force equals the resistive force. The drag polar is dependent on aircraft specific knowledge (e.g., frontal surface area, weight) and atmospheric data (e.g., air density ratio). The detailed equations are derived in Appendix A.

The qualitative knowledge of aircraft performance is dependent on the interactions between lift, drag, thrust, and weight. Lift and thrust are positive forces while weight and drag are resistive forces. Drag has two components: parasitic and induced drag. Parasitic drag is a pressure force that is proportional to the air density ratio and velocity. Thus as velocity increases, parasitic drag increases. As air density increases (corresponding to decreasing altitude), parasitic drag increases. Induced drag is dependent on lift and is inversely proportional to velocity and air density ratio. Thus as lift increases (more wing area is exposed to the wind) induced drag increases. The point on the drag polar where parasitic drag equals induced drag is called the 'maximum lift to drag ratio.' At this point, a velocity is obtained that minimizes drag which corresponds to a minimum fuel flow. However, aircraft wish to fly at speeds greater than this velocity for several reasons. First, there are stability considerations which are not modeled by equations in this work. Slight perturbations from equilibrium at the minimum drag point may require constant throttle adjustments which tend to be annoying to the pilot. More importantly, for jet aircraft, fuel flow consumption is proportional to thrust. Thus, the velocity for maximum range is greater than the velocity for maximum endurance. We define the maximum endurance velocity as the lower limit for an aircraft in a high altitude sector.

There are six states or unique equilibrium conditions for an aircraft.

1. steady state level flight (constant velocity)
2. steady state climb
3. steady state descent
4. level acceleration
5. level deceleration
6. steady state turn

The typical aircraft in controlled airspace has the enroute intent which means it desires to maintain the level state. The other states are transitional states. The relationship between the four aircraft forces and the states is shown in Figs. 16 and 17 (p. 68, 69).

3.0. Interfacing the Domain Equations

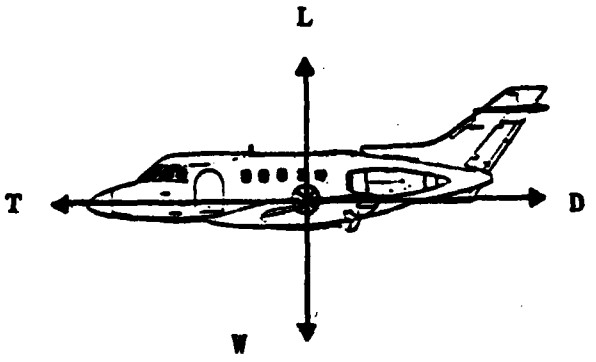
The equations of Figs. 16 and 17 are derived for an aircraft body axis. The reference frame is related to a earth-surface reference frame by the pitch and roll angles. An earth-reference frame is the frame used by controllers. The force equations can be decomposed into two parallel one-dimensional representations, each of which is a specific instance of the abstracted representation in Fig. 14 (p. 62).

For the remaining portion of this chapter, I will deal with an aircraft that is in the level state. The two new representations will be called the 'horizontal aircraft level' and the 'vertical aircraft level' and, for brevity, will be referred to as the 'HORZ-LVL' and the 'VERT-LVL.' The interface is said to be equation dependent because the state dependent force equations and semantic knowledge about domain forces define the linkages to the abstracted level. Additionally, the influences to a domain force are defined via a symbolic series expansion of that force's dependent variables. The 'level state' HORZ-LVL and VERT-

Level Flight

$F_x = T - D = 0$

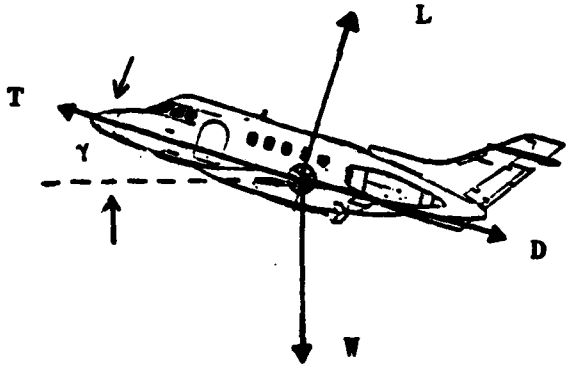
$F_y = L - W = 0$



Climb

$F_x = T - W \sin \gamma$

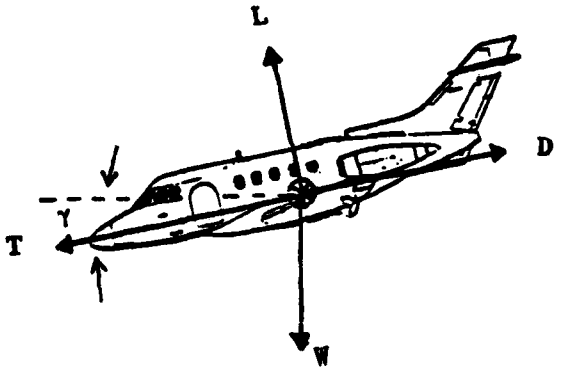
$F_y = L - W \cos \gamma$



Descent

$F_x = T - D + W \sin \gamma$

$F_y = W \cos \gamma - L$

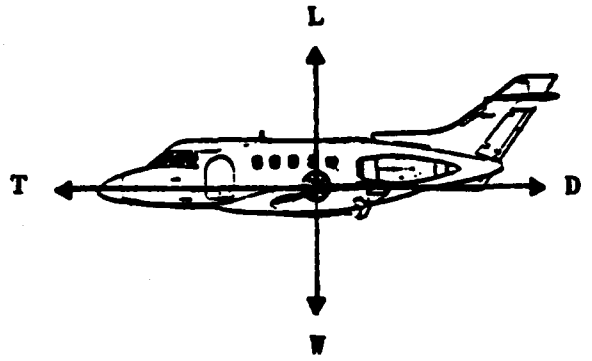


Aircraft Force Equations
Figure 16

Acceleration

$F_x = T - D > 0$

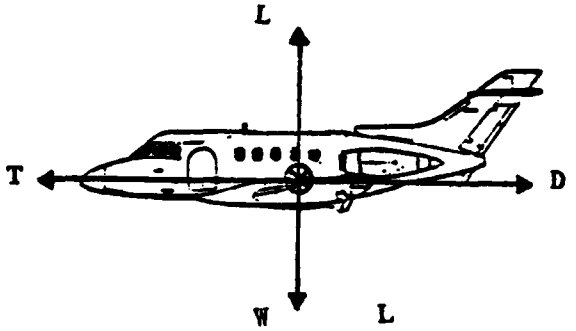
$F_y = L - W = 0$



Deceleration

$F_x = T - D < 0$

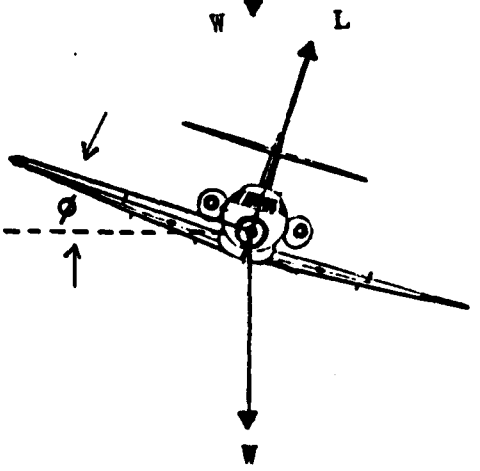
$F_y = L - W = 0$



Turn

$F_x = T - D > 0$

$F_y = L - W > 0$



Aircraft Force Equations (Continued)
Figure 17

LVL will now be derived.

First consider the creation of the HORZ-LVL. The coefficients of the T (thrust) and D (drag) error term indicate whether they are positive or negative influences. Since the error terms are constants, the linkages can be permanently defined. I will claim that a quantity is a primary influence if it is the instance of a concept in the abstracted level and that abstracted concept is a primary influence in the abstracted level. Otherwise, an influence is considered secondary. Without additional semantic information, the heuristic provides a 'best guess' as to which new variables are likely to cause a change and which do not. The intent is to limit the search level when constraints are propagated forward and backward. At the naive physics level, a positive force can either be considered a propulsive or an enabling force. It is the responsibility of the domain variable to define semantics which allow a correct specification. For instance, by definition, thrust can be a propulsive force, while lift must be an enabling force. So thrust is labeled as an instance of propulsive force. Now consider drag. Drag consists of two components: induced drag and parasitic drag. Both are positive influences of drag since $D = D_p + D_i$ and drag is a resistive force which is a positive influence. The equation for each is derived in Appendix B. Consider a symbolic representation of parasitic drag which will be used to illustrate how its influences are derived.

```
(frame drag a node w
    (name = drag)
    (components = (Dp Di))
    (level = horz-level)
    (parent = (Fres)))
```

```
(frame Dp a node w
    (name = parasitic-drag)
    (influenced-by = ((primary+ den vh) (secondary+ f)))
    (level = horz-level)
    (component-of = (drag))
    (equation = (times .0033898 f (expt vh 2) den)))
```

The influences of parasitic drag's independent variables are based on two facts: the sign of the first error coefficient when expansion is accomplished around that variable's nominal value and evidence of that variable's existence as a concept at the abstracted level. Consider 'v' which represents horizontal velocity. The series expansion is shown below.[3]

```
-> (series-expansion 'Dp 'v)
   (times .0033898 (expt f 1) (expt d 1)  2.0 (expt v 1))
```

Since the coefficient is positive and velocity is a concept defined in the abstract level, which is itself a primary influence, horizontal velocity becomes a primary positive influence of parasitic drag. Density is a positive influence dependent on yet-to-be defined knowledge in the VERT-LVL. Frontal surface area is a secondary positive influence because a concept for surface area does not exist in the abstracted level. This is reasonable because we have only represented, at the abstract level, those physical concepts which usually affect dynamic equilibrium. In most contexts, frontal surface area will not change. However, there are exceptions. For instance, one may be comparing the relative affects of aircraft shape or one may wonder what might happen if the bomb bay doors are opened (thus increasing the frontal surface area). Procedures which

---

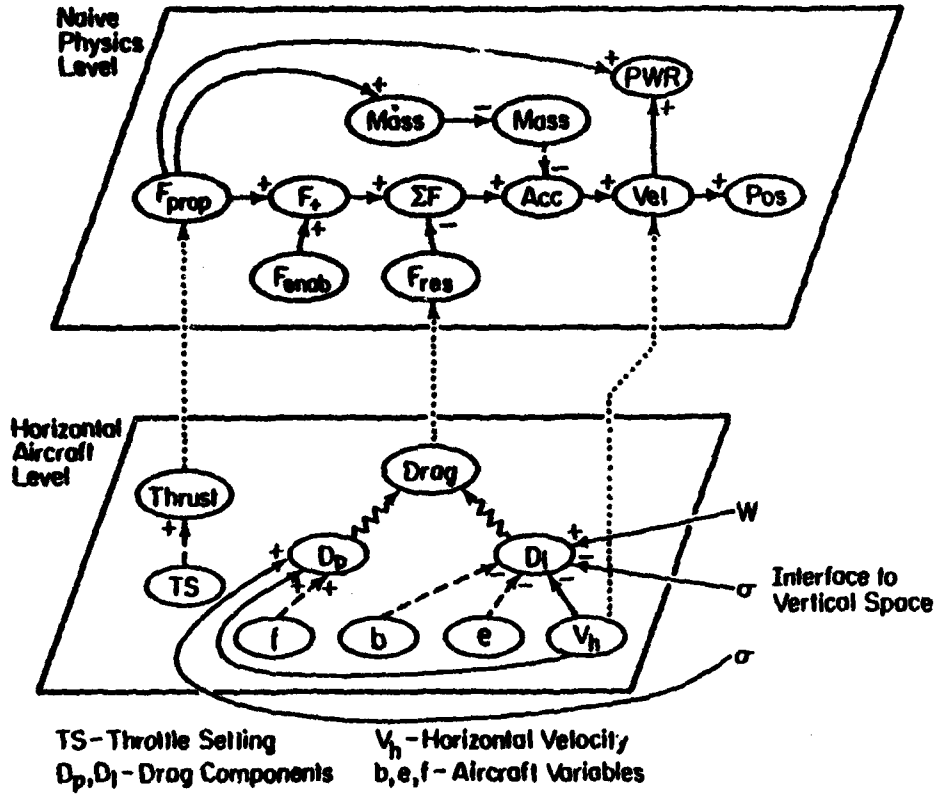
[3]Admittedly, the way in which the equations are coded simplifies the series expansion.

utilize these levels must indicate the contexts in which secondary linkages become active. The completed interface is shown in Fig. 18 (p. 73).

The representation of the VERT-LVL is accomplished in the same fashion. Lift is an enabling force that is positively influenced by the horizontal velocity, the air density, the lift curve slope, and the angle of attack. Weight is a resistive force that is positively influenced by mass and negatively influenced by gravity. Sometimes influences must be derived recursively. For instance, air density is a positive influence on lift and parasitic drag, but we have not yet specified whether it is a primary or secondary influence. The equation for air density is given in Appendix B and decreases as altitude increases. Thus, air-density is negatively influenced by altitude. Altitude is vertical position. Position can be a primary influence of resistive force (e.g., spring force). We infer that air density is a primary influence. The representation of both aircraft spaces and the abstracted space is shown in Fig. 19 (p. 74).

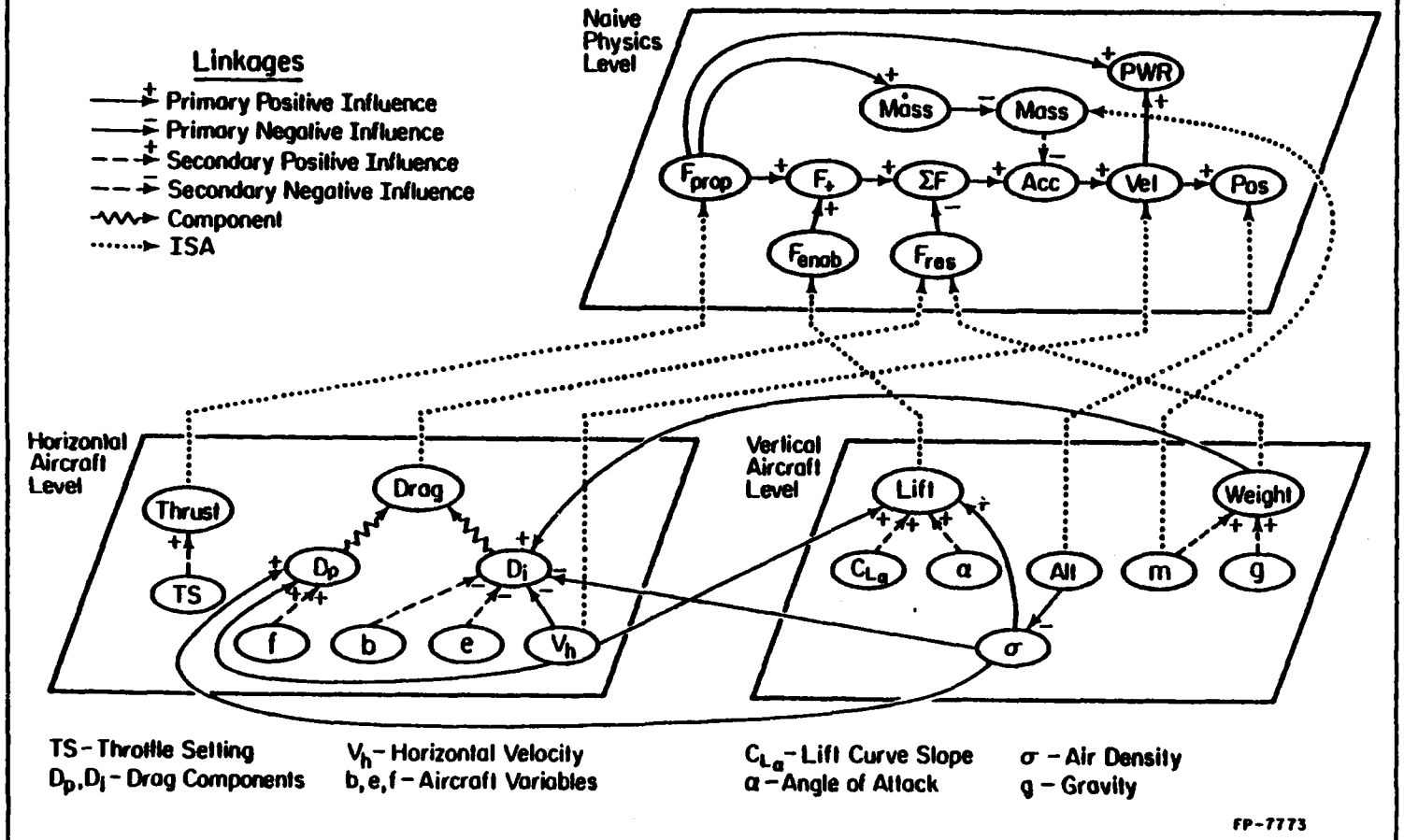
4.0. Interfacing Conceptual Knowledge

At the conceptual level, the computer has knowledge about goals and plans to achieve those goals. There must also exist knowledge about plan evaluation. In the case of collision avoidance, simple critics that computed relative position between aircraft were used. I made extensive use of a network representation that defined conflicts. That representation is roughly analogous to the diagram developed in this chapter. Part of the conceptual knowledge interface is the specification of procedures that can evaluate plans using the diagrams. Two additional aspects of



Interface to the Horizontal Aircraft Level

Figure 18



Interface to the Aircraft Levels
Figure 19

the conceptual knowledge are important. The first is a form of envisioning. The controller's envisioning process requires him to visualize the future states and state changes (inferred from flight plans and control directives) that will satisfy an aircraft's goals. For example, a controller may direct a climbing aircraft to maintain its altitude until overhead traffic passes. The decision whether the aircraft should be allowed to continue its climb is dependent on that aircraft's future goals and the environment in which the aircraft finds itself. In this work, I have considered two goals (safety and fuel efficiency) and have adopted a rigid goal resolution strategy: safety over fuel efficiency.

The second important aspect can be considered heuristic advice to the domain level representations, and concerns the magnitude of the forces in the force equations. I assume that thrust is solely a function of throttle setting. In reality, it is also a function of thermal efficiency (which in turn is a function of air temperature) and aircraft velocity.[4] Since these two aspects counter one another, they are usually disregarded in the aeronautical literature (Perkins and Hage [50], Nelson [47]). I assume that climbs and accelerations are accomplished at full throttle and that descents and decelerations are accomplished at idle throttle. This is a simplification of the real world. Typically, at least one engine must supply thrust to drive the electrical generators. Additionally, while the assumptions about full and idle throttle are perfectly legal ways to fly, individual or company airline

[4]Thrust is dependent on the fuel flow rate and the inlet air mass. Temperature decreases as altitude increases, thus air is less dense, so air mass decreases. But this is compensated by a 'ram effect.' Since the air is less dense, the aircraft's true velocity increases, which increases the air mass available to the engine inlet.

procedures may be different.[5] Wesson [64] relates that human controllers recognize the subtle differences in flying technique among airlines (and even pilots) and factor that knowledge into their plans.

The procedure used to evaluate fuel efficiency involves computing the fuel used with and without a plan. The difference is that extensive use will be made of qualitative knowledge. If one wishes to verify that less fuel is used, but is not concerned with how much less fuel is used, then one need only verify that less propulsive force is required. Additionally, by using the diagrams developed based on the aircraft performance equations, the computer understands the qualitative knowledge contained in the procedure. For instance, if the goal is to verify that less propulsive force is required, then one may try to verify that less resistive force is encountered. Examples will now be given to illustrate qualitative reasoning.

5.0. Examples

In this section, I will present an example that illustrates the uniqueness of the qualitative reasoning capability. First, I claimed that the reasoning capability supports bidirectional reasoning. That is, reasoning could be accomplished from qualitative knowledge to quantitative knowledge and vice versa. Consider the example which was used in the beginning of the chapter about a climbing aircraft. Is the plan fuel efficient? It is if less fuel is used during the remaining flight. The answer can be arrived at qualitatively if each state change requires

---

[5] My assumptions are suggestive of the techniques employed by B-737 pilots that fly for the smaller carriers like Piedmont or Air California.

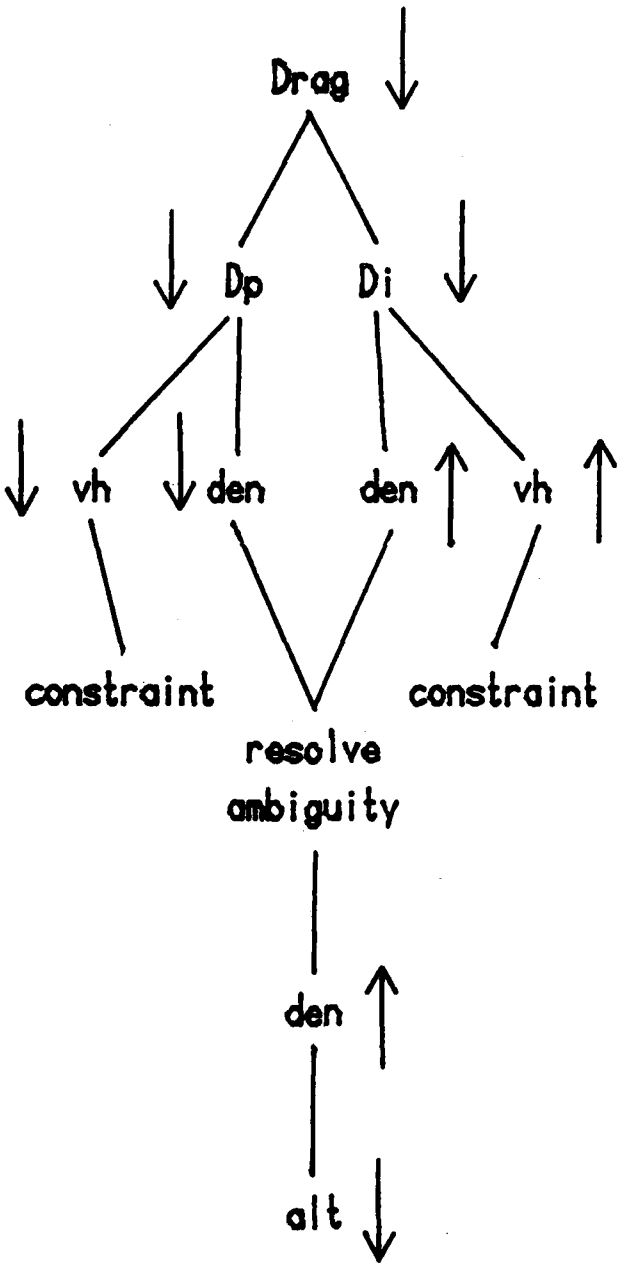
less fuel than the original state and if the time to achieve the final state is not extended. There are two new states added by the plan: the climb state and the level flight state (at a higher altitude). It is obvious that fuel flow rate increases during the climb because the climb is accomplished at full throttle. Now the problem becomes one in which it must be decided if the fuel flow rate at the higher altitude is increased from the original state. If it also increases, a contradiction is reached and it can be concluded that the command is not fuel efficient. Tracing through the diagram, the parameter that changes is air density which has a positive influence on induced drag and a negative influence on parasitic drag. It is observed that if drag decreases, then resistive force decreases; hence propulsive force will decrease. But an ambiguity must be resolved to determine which drag component has the greatest influence. Suppose, that density has a greater influence on parasitic drag. Then drag decreases so fuel flow rate also decreases. It becomes apparent that a quantitative method must be applied. The transition time of each transition can be computed. The times are available from the equilibrium demons activated when each state change is simulated. Fuel used during the climb is then easily found since maximum thrust was used. The time at the next level state is the remaining flight time. The fuel used is computed from the required thrust which equals the drag.

Consider two cases involving a heavy B-747. In the first case the aircraft is above its optimal altitude so a climb would be fuel inefficient. In the second case it is above its optimal altitude, so the climb would be fuel inefficient. The example was chosen because novice

controllers are often suprised that aircraft must be descended to reach a fuel efficient cruise altitude. The diagram in Fig. 20 (p. 79) illustrates the computer search.

The space representations make the influences of each domain variable explicit. This is the essence of the computer's understanding. For instance, a human controller may be suprised that the above aircraft would have to descend to reach a more fuel efficient altitude. After all, their hueristic is to climb aircraft. An explanation can be rendered that illustrates how quantitative knowledge can be expressed qualitatively. So, the computer can trace through the diagram showing that density affected both parasitic and induced drag, and in this case, parasitic drag was predominant. Therefore, drag decreased as altitude decreased.

A controller may be confused by an answer that incorporates excessive domain dependent vocabulary. A summary explanation may be sufficient. The same procedure that created an explanation in an aircraft space can be used to create an explanation in the abstracted space.



Qualitative Simulation with a HB747
Figure 20

CHAPTER VI

APPLICATIONS OF QUALITATIVE REASONING

In this chapter, the qualitative reasoning approach will be shown to be useful in three distinct problem areas. First, it will be shown to be useful for the justification of the expert system collision avoidance plans. The justification is based on explaining a plan's impact on aircraft performance. Second, the qualitative reasoning approach is useful for justifying the plans of another, perhaps more knowledgeable, expert. In this context, plan justification is used for the interpretation of advice, a necessary step towards advice-initiated learning. I have previously described the role advice-initiated learning plays in the human controller training process. Third, it is shown that the reasoning approach and equation-based structural diagrams are applicable in another domain. For this application, the automobile is used. The naive physics knowledge remains unchanged and new heuristic and mathematical knowledge is specified.

1.0. Justification of Air Traffic Control Plans

I assume that at a conceptual level, goals are recognized and plans created. Justification means that well-founded reasons can be put forth that show the plan is reasonable in the particular context. The capability developed here then allows the justification of aircraft performance goals. While I have focused on fuel efficiency in the previous chapter, in this section I will show the applicability of the reasoning process to many performance related goals. The semantic network of equational influences serves both as a vehicle for qualitative simulation and a

data base with which questions can be answered.

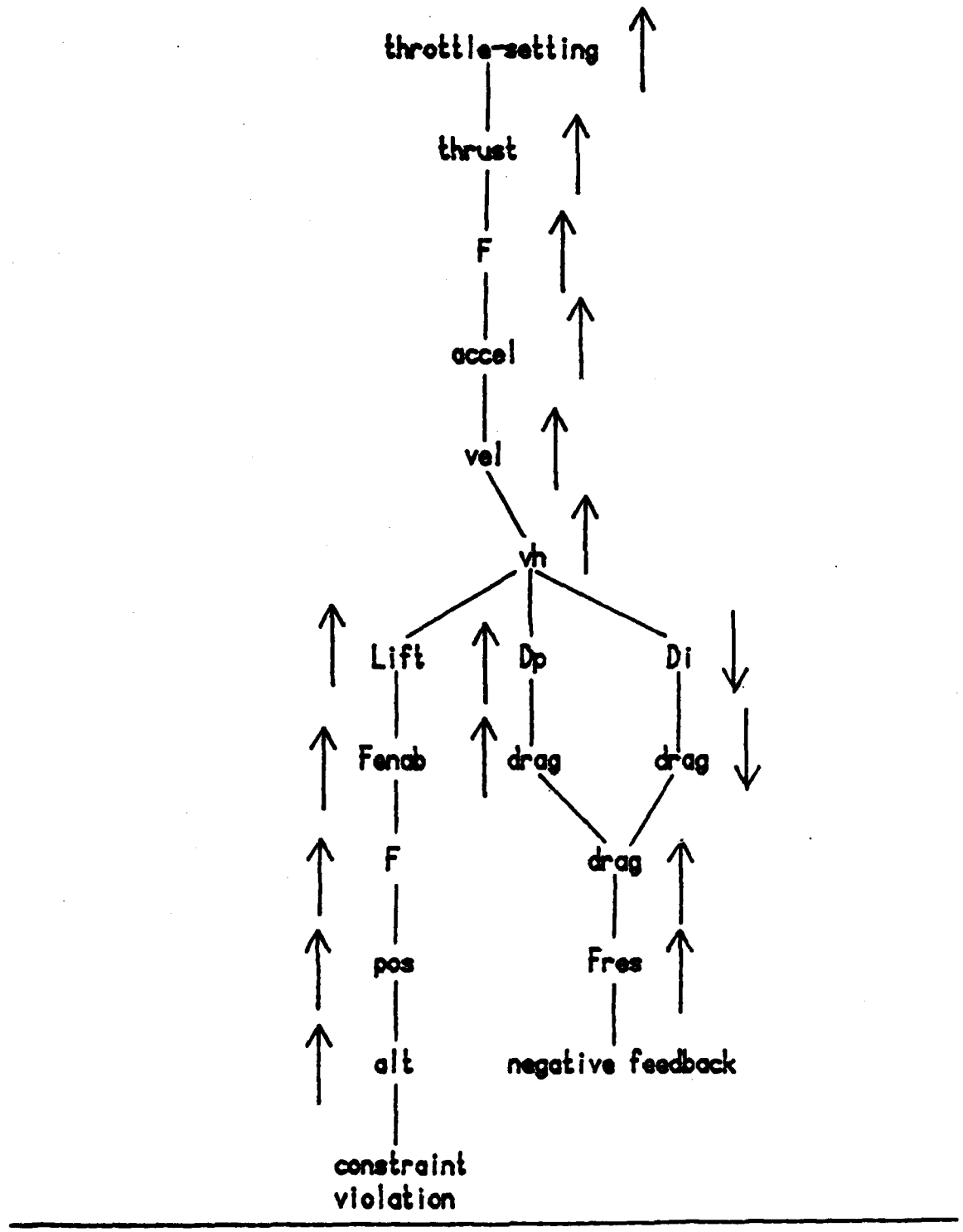
Consider a plan that requires an aircraft to increase its speed. Unless explicitly stated, the plan implies that a constant altitude be maintained. An important aspect of air traffic control is that the controller never invoke a plan that is not executable by the aircraft. Given the plan to increase speed, how can the controller be sure that (1) the aircraft can increase its speed and (2) maintain a constant altitude? The computer can obtain the correct answers from the semantic network of performance equations. An increase of airspeed occurs when its parent, velocity, is increased. An increase in velocity is caused by an increase in acceleration which is, in turn, caused by an increase in positive force or a decrease in drag. Drag cannot decrease because its influences are constrained (velocity is to be increased and density is constrained to 'nc by altitude). Thus, positive force must increase which implies an increase in throttle setting.

If the throttle setting is at 100%, then the aircraft cannot increase its airspeed. When throttle setting is increased, that value can be propagated forward to find its influences. There is then a tendency for the increased airspeed to influence lift. Lift increases which tends to make the aircraft increase its altitude. But this violates a constraint because altitude is constrained not to change. A search is made for influences that can counteract the positive influence to lift by airspeed. There are no primary influences. Therefore, a search is made for secondary influences. There are three: lift curve slope, wing area, and angle of attack. An intelligent choice from this set of variables requires the use of context related knowledge. For instance, the

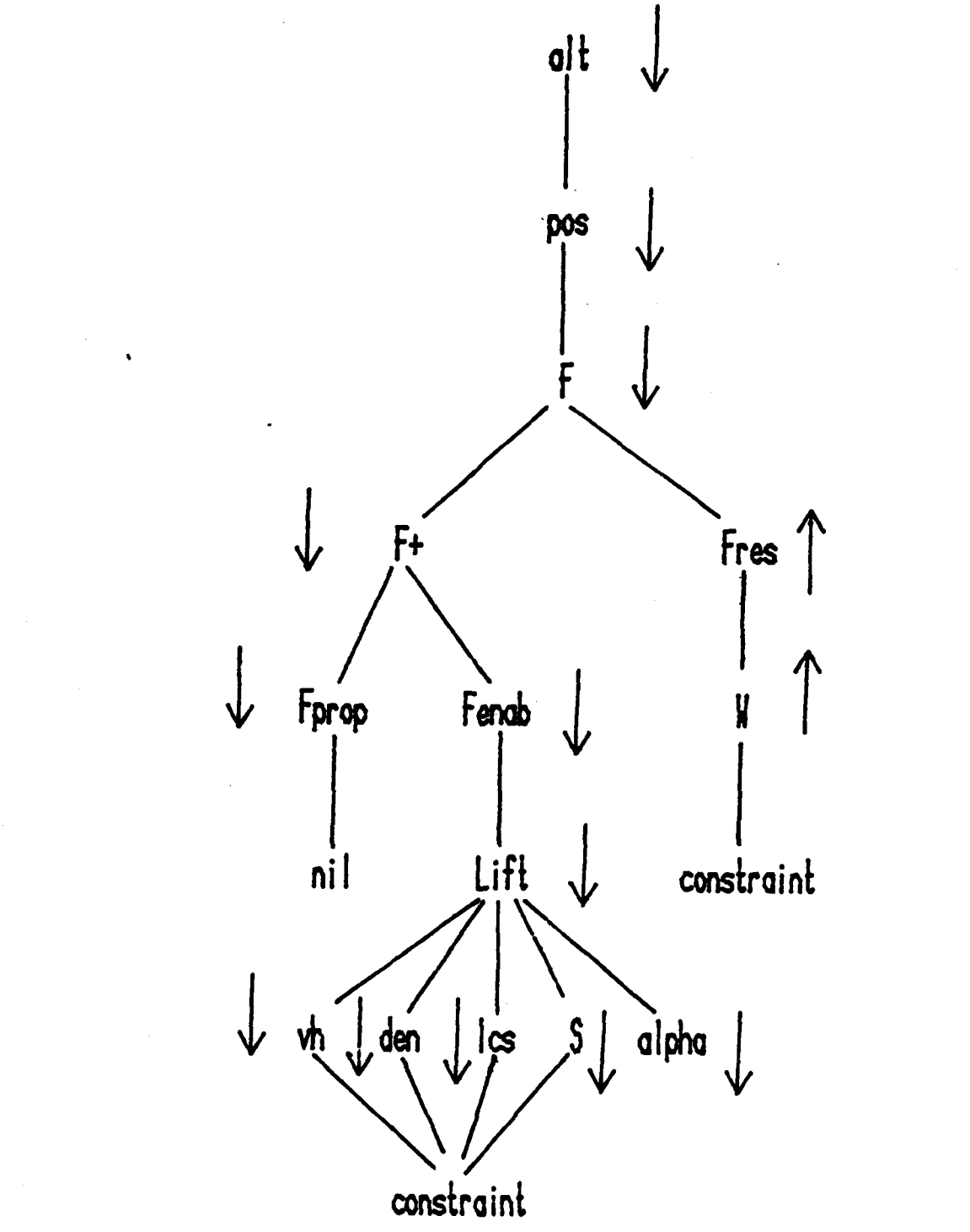
knowledge that the wing area is constant for a given aircraft must be represented in the frame for the node that represents wing area. Similarly, there must be represented the knowledge that the lift curve slope is constant for the context of high altitude flight and that the lift curve slope increases in the context of landing (when the flaps are deployed). Angle of attack is pilot controllable and thus is the correct answer. The reasoning process is illustrated in the search tree of Fig. 21 (p. 83).

The similar reasoning process is applied when the aircraft is commanded to decrease its speed. Throttle setting is decreased and angle of attack is increased. The increased angle of attack illustrates a constraint known as a limit point. An aircraft maintains lift at lower airspeeds by increasing the angle of attack. This is possible because the lift curve slope is a positive constant for small angles of attack. But when the angle of attack becomes too large, the lift curve slope becomes negative and the aircraft stall. Thus the limit point procedure checks the angle of attack value whenever the value is changed to 'incr and constrains the value to 'nc if the value is considered excessive. I should point out that a pilot may fly with an angle of attack meter. His use of the meter is similar to the approach taken here. He may occasionally glance at it and only take responsive action to its reading when the indicator is increasing and in the 'red' area.

In the previous chapter, I described the reasoning process as pertaining to a climb command. Consider several cases where it is not clear if the aircraft should climb or descend. In the first instance, an aircraft may be more fuel efficient at a lower altitude. Consider an air-

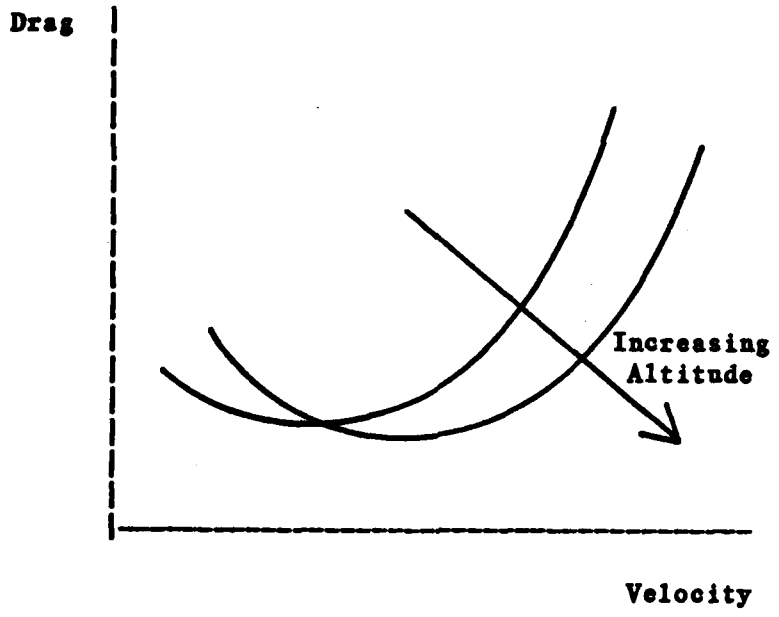


Increasing Airspeed with an Altitude Constraint
Figure 21



Increasing Airspeed with an Altitude Constraint (Cont'd)
Figure 21

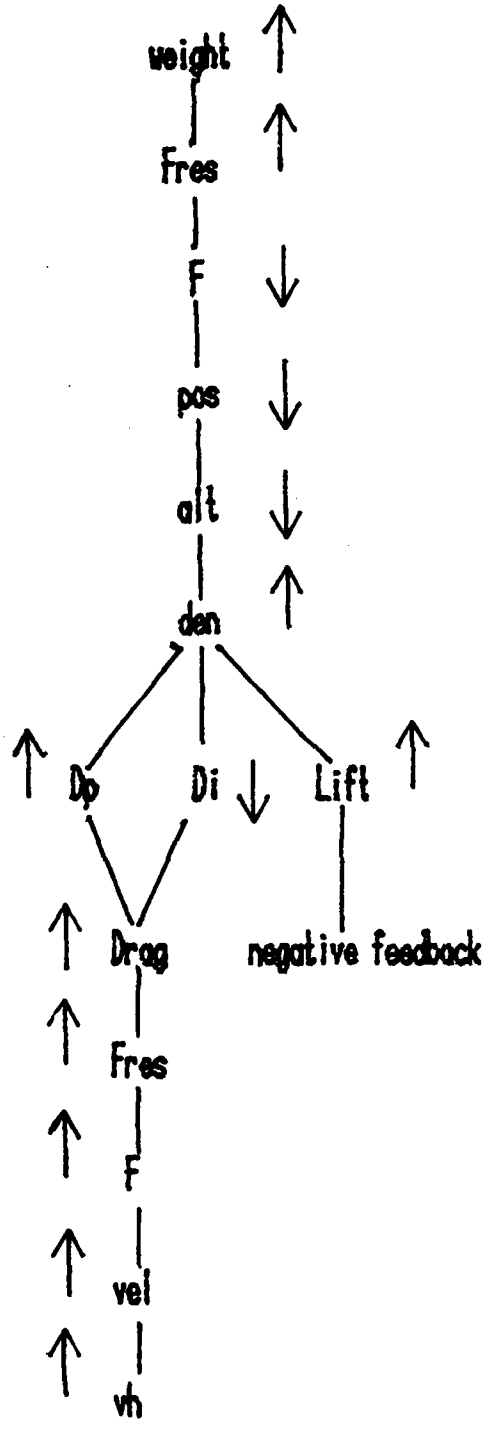
craft in a time metered environment (velocity is constrained). Drag then varies inversely with air density as shown in Fig. 22 (p. 85). If the aircraft is above the minimum air density, then it should decrease its altitude. As another example, consider the weight of an aircraft. Normally, weight changes very slowly. As weight decreases the aircraft wishes to climb to a more fuel efficient altitude. If velocity remains constant, there comes a point when the aircraft needs to descend. Now consider a military aircraft like a C-5A or a B-52G that receives



Drag as a Function of Air Density
Figure 22

several hundred thousand pounds of fuel during an airborne refueling maneuver. Assume that this is done during an exercise in controlled airspace. The computer has not been programmed to handle the case when an aircraft's weight increases. The resultant aircraft performance is found by qualitative simulation. When weight is increased, the aircraft will descend unless lift is increased. Assume that lift is increased implying that altitude is constrained. This implies that velocity must increase which can be traced back to an increase in throttle setting. Thus, the aircraft is now more fuel inefficient. The problem of deciding if the aircraft should climb or descend is fairly straightforward. A trace is shown in Fig. 23 (p. 87).

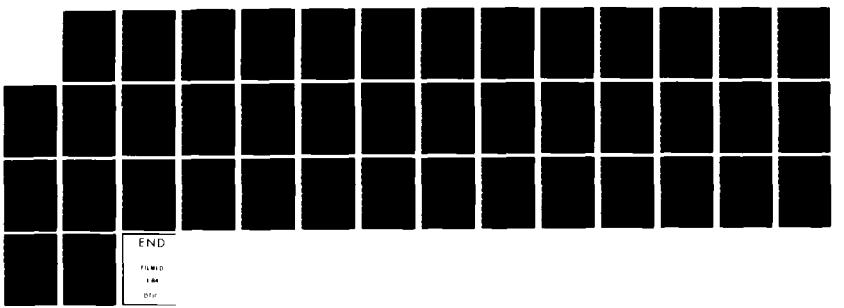
The final example in this section illustrates the use of qualitative reasoning in an emergency situation. Consider the case when an engine fails. The computer should be able to simulate the effect of a lost engine and deduce the performance degradations. The equation originally given for thrust in Chapter V must now be replaced with a more detailed description. Thrust is a function of both the throttle setting and the number of engines. For purposes of discussion, consider a DC-10 which is represented in the aircraft data base by the following frames. The knowledge contained in the DC-10 frame is used when exact computations are required to resolve ambiguities. For instance, there is an ambiguity in drag. Exact computations must be performed for parastic and induced drag, which require instantiation of the variables altitude, weight, and frontal surface area.

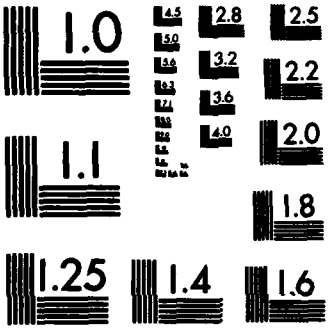


What Happens When Aircraft Weight Is Increased
Figure 23

AD-A135 444 QUALITATIVE REASONING IN AN EXPERT SYSTEM FRAMEWORK(U) 2/2
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
S E CROSS MAY 83 AFIT/CI/NR-83-70D

UNCLASSIFIED F/G 5/1 NL





```
(frame dc10a a commercial-jet-aircraft with
       (chord = 155.0)                ; feet
       (aspect-ratio = 6.8)           ; chord**s/wing-area
       (wing-area = 3861.0)           ; feet**2
       (weight-empty = 245000.0)      : pounds
       (weight-takeoff = 455000.0)    ; pounds
       (engine-type = CF6-6D)
       (engine-number = 3))

(frame CF6-6D  a turbofan with
       (thrust-available = 9120.0)    ; pounds
       (spfc = .61))                  ; lbs/hr per lb-thrust
```

From the structure, it is apparent that when the engine number decreases, that the thrust also decreases. Possibly, thrust could be compensated by an increase of throttle setting. The controller should be able to predict the impact of aircraft performance in this scenario. The aircraft can continue at the present altitude if there is sufficient thrust to maintain a velocity above the stall velocity. When altitude is constrained not to change, the decreased value of thrust is propagated in the network. For this particular example, the aircraft can continue its flight.

2.0. Advice Interpretation

I define advice interpretation to be the justification of a plan rendered by another expert. This is a necessary component of learning, because to truly learn something, one must first understand the use of that knowledge in the context to which it applies. I will illustrate advice interpretation with two examples.

The first example introduces a new concept called wind. The controller understands aircraft and plans in a ground referenced system, but aircraft fly relative to moving air masses. Thus, understanding wind

and its impact on aircraft performance is a critical aspect of air traffic control. The advice may be communicated in several different ways. For instance, it may suffice to tell someone that a 'headwind' decreases velocity and a 'tailwind' increases velocity. The advice is interpreted as the statement "a headwind is a negative influence on velocity and a tailwind is a positive influence on velocity." These wind concepts then are easily added to the horizontal aircraft space of the aircraft domain.

```
(frame headwind a node w (influences = (vh)))
(add-to-equation-for vh (- headwind))
(frame tailwind a node w (influences = (vh)))
(add-to-equation-for vh (+ tailwind))
```

The affects of wind can be derived from the semantic network. For instance, a headwind is a negative influence on velocity. If the goal is to maintain velocity, then the thrust must be increased.

A second example concerns fuel efficient descent procedures. Recent research (Stengel and Marcus [61]) has sought to show the benefits of idle descents at $V_{L/D}$. The heuristic that airline operators and controllers operate under is that it is more fuel efficient to remain at a fuel efficient cruise altitude for as long as possible and then accomplish a maximum rate descent to a terminal sector. Controllers also prefer this mode of operation because the safety task is simplified when aircraft are not changing altitude. An algorithm proposed for a C-141 by Stengel and Marcus [61] and tested by NASA on a B-737 (Knox [36]) requires an arrival aircraft to initiate a descent approximately 40 miles before a 'normal' descent point and fly at $V_{L/D}$. The descent is to be accomplished at idle thrust with pitch control used to maintain airspeed. An

equation-based structure must be created that illustrates the algorithm.

Consider the equations of motion for a descending aircraft. The force equation along the aircraft thrust line is:

$$T - D + W \sin \gamma = 0 \tag{6.1}$$

The force equation perpendicular to the thrust line is:

$$- L + W \cos \gamma = 0 \tag{6.2}$$

where γ represents the aircraft pitch angle. A new structure can be created from these equations. As before, thrust, lift, and drag are propulsive, enabling, and resistive forces, respectively. The difference is that the terms (W sin gamma) and (W cos gamma) are positive forces and lift is a resistive force. Conceptual knowledge about aircraft must now be considered. When an aircraft is descending, thrust is set to idle. The important aspect of this algorithm is that the velocity is constrained to the 'maximum endurance' airspeed. An obvious question is "How can the velocity be controlled?" Consider the effect of γ. The influence of γ is seen from the series expansion of each equation about a nominal value γ_o.

$$\Delta F_x = W (\sin \gamma_o + \Delta\gamma \cos \gamma_o) \tag{6.3}$$

$$\Delta F_y = - W (\cos \gamma_o - \Delta\gamma \sin \gamma_o) \tag{6.4}$$

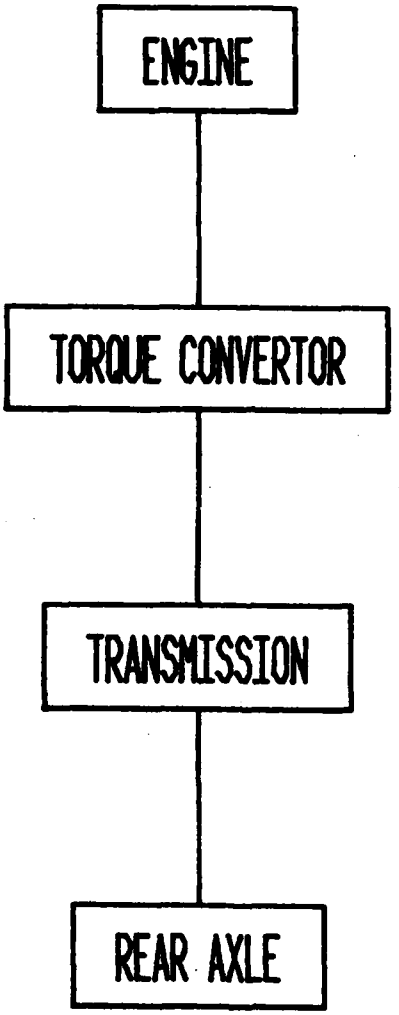
Small angle approximations are used for the sine and cosine of $\Delta \gamma$. Pitch angle is defined clockwise from the horizon. The influences of perturbations to the pitch angle are now easily derived. When pitch is increased, the influence to F_x due to weight increases, thus the airspeed increases. From the representation, the use of pitch as a

speed control is understood.

3.0. Reasoning About Automobile Performance

In this section, I will demonstrate that the same reasoning approach is applicable to automobiles. Specifically, I intend to show that the computer can construct a semantic network based on simplified equations that model a gasoline combustion engine that is used to drive a real axle powered car. These equations are interfaced with the resistive forces of an automobile: tire drag and aerodynamic drag. My objective is to demonstrate the computational aspects of a new representation. However, I do not intend to argue its utility in the automobile domain. Some interesting applications may be for use by a designer or in an onboard computer used in a diagnosis task. Beachley [6] describes how an onboard computer could be used to implement a continually variable automatic transmission.

The propulsive force of an automobile is conceptually more complicated than jet engine. The block diagram of Fig. 24 (p. 92) illustrates the components of an automobiles propulsive force. The force at the rear wheels is dependent on the driveshaft torque and the radius of the tires. The driveshaft torque is dependent on the torque convertor output and the transmission gear ratio and efficiency. The torque convertor output is a function of the input engine torque and a term known as the torque ratio. The torque ratio is a function of the speed ratio, the ratio of the driveshaft angular velocity to the engine angular velocity. The engine torque if dependent on the engine speed and the input size factor. I have assumed that the engine speed is solely a function of the throttle setting (in reality it is a function of the carburetor which



Components of Automobile Propulsive Force
Figure 24

controls the air-fuel mixture). The equations are shown in Appendix D.

An automobile has three predominant resistive forces: drag, tire friction, and weight (if not on a level path). The drag equation is predominantly a pressure force. Tire friction is dependent of the coefficient of friction between the tires and the road surface and the weight of the automobile. Tire friction decreases with increasing velocity. The resistive force due to weight is related to the sin of the climb angle.

The equations can be represented by nodes. The same naive physics representation is used. One example is given to illustrate that the qualitative reasoning approach is domain independent. One's conceptual knowledge about the automobile probably contains common sense knowledge like 'tire radius does not change.' This data is represented by a constraint in the tire radius frame. Assume that the car begins a climb. Can the automobile maintain its present velocity? Does it have to downshift? The results of the qualitative simulation are shown in Fig. 25 (p. 94). A few of the nodes are shown below.

```
(frame FRW a node w
    (level = auto)
    (name = rear-wheel-force)
    (equation = (times TRW (expt r -1)))
    (parent = (Fprop)))

(frame TRW a node w
    (level = auto)
    (name = drive-shaft-torque)
    (equation = (times TCO gr ntr)))
```

The example illustrates the backward search portion of the qualitative reasoning program. Assume that speed is to be increased because of the desire to maintain a constant velocity when climbing. The conceptual knowledge of the automobile constrains the values of tire radius and the coefficients of drag and friction.

The two functions back1 and back2 concentrate on primary and scondary links respectively.

```
-> (backtrace-through-nodes 'spd 'incr)

1 <Enter> back1 (spd incr)

  since no influence to speed
  propagate the value to the parent

|2 <Enter> back1 (vel incr)
| 3 <Enter> back1 (accel incr)
| |4 <Enter> back1 (F incr)
| | 5 <Enter> back1 (F+ incr)
| | |6 <Enter> back1 (Fprop incr)

  propagate to the instance, rear wheel force

| | | 7 <Enter> back1 (FRW incr)
| | |6 <EXIT>  back1  nil
| | |6 <Enter> back1 (Fenab incr)
| | |6 <EXIT>  back1  nil
| | 5 <EXIT>  back1  nil
| | 5 <Enter> back1 (Fres decr)
| | |6 <Enter> back1 (D decr)
| | |6 <EXIT>  back1  nil
| | |6 <Enter> back1 (TD decr)
| | |6 <EXIT>  back1  nil
| | |6 <Enter> back1 (FW decr)
| | |6 <EXIT>  back1  nil
| | 5 <EXIT>  back1  nil
| |4 <EXIT>  back1  (Fres)
| 3 <EXIT>  back1  nil
|2 <EXIT>  back1  nil
1 <EXIT>  back1  nil
```

Automobile Example
Figure 25

```
  no primary influences applicable
  so look for secondary influences

1 <Enter> back2 (spd incr)
|2 <Enter> back2 (vel incr)
| 3 <Enter> back2 (accel incr)
| |4 <Enter> back2 (F incr)
| | 5 <Enter> back2 (F+ incr)
| | |6 <Enter> back2 (Fprop incr)
| | | 7 <Enter> back2 (FRW incr)
| | | |8 <Enter> back2 (TRW incr)
| | | | 9 <Enter> back2 (TCO incr)
| | | | |10 <Enter> back2 (TE incr)
| | | | | 11 <Enter> back2 (NE incr)
| | | | | |12 <Enter> back2 (ts incr)
| | | | | |12 <EXIT>  back2  (incr ts)
| | | | | 11 <EXIT>  back2  nil
| | | | | 11 <Enter> back2 (KI decr)
| | | | | |12 <Enter> back2 (SR decr)
| | | | | | 13 <Enter> back2 (NCO decr)
| | | | | | |14 <Enter> back2 (gr decr)
| | | | | | |14 <EXIT>  back2  (decr gr)
| | | | | | 13 <EXIT>  back2  nil
| | | | | |12 <EXIT>  back2  nil
| | | | | 11 <EXIT>  back2  nil
| | | | |10 <EXIT>  back2  (KI)
| | | | |10 <Enter> back2 (TR incr)
| | | | |10 <EXIT>  back2  nil
| | | | 9 <EXIT>  back2  nil
| | | |8 <EXIT>  back2  nil
| | | 7 <EXIT>  back2  nil
| | |6 <EXIT>  back2  nil
| | |6 <Enter> back2 (Fenab incr)
| | |6 <EXIT>  back2  nil
| | 5 <EXIT>  back2  nil
```

Automobile Example (Continued)
Figure 25

```
| | 5 <Enter> back2 (Fres decr)
| | |6 <Enter> back2 (D decr)
| | |6 <EXIT>  back2  nil
| | |6 <Enter> back2 (TD decr)
| | |6 <EXIT>  back2  nil
| | |6 <Enter> back2 (FW decr)
| | |6 <EXIT>  back2  nil
| | 5 <EXIT>  back2  nil
| |4 <EXIT>  back2  (Fres)
| |4 <Enter> back2 (mass decr)
| | 5 <Enter> back2 (mass-rc incr)
| | 5 <EXIT>  back2  nil
| |4 <EXIT>  back2  (mass-rc)
| 3 <EXIT>  back2  (mass)
|2 <EXIT>  back2 nil
1 <EXIT>  back2 nil


*infl-changes*
((incr ts) (decr gr))
```

implies that one can increase the throttle setting or downshift

Automobile Example (Continued)
Figure 25

CHAPTER VII

SUMMARY AND CONCLUSIONS

The major contribution of this research is the development of a qualitative reasoning capability that allows the computer to understand domain equations. A qualitative reasoning capability facilitates the integration of mathematical knowledge with a heuristic knowledge base. The reasoning process is useful for the justification of expert plans and the interpretation of another expert's advice. The approach is based on the propagation of qualitative values in a structure which the computer constructs from the domain equations. The domain equations are interpreted at an abstract level.

Two contributions are claimed in the realm of air traffic control expert systems. Representing knowledge in frames is an improvement over previous production systems [52, 64]. The manner in which problems are defined (in terms of the conflict network) allow many goal interactions to be easily recognized and more intelligent problem solving strategies specified.

The theoretical contributions of the research are:

(1) Domain equations are interpreted in terms of a naive physics representation of Newton's laws as applied to one-dimensional motion thus abstracting the influences inherent in the equations.

(2) The computer constructs its own representation based on a symbolic series expansion.

(3) The reasoning is bidirectional.

The naive representation of Newton's laws is an explicit statement of the designer's understanding of Newtonian mechanics applied to one-dimensional motion of a mechanical system. The variables (e.g., force, velocity) are related by influence links. An influence link defines how one variable changes in relation to another changing variable. The influence links are assigned as primary or secondary based on their relationship to nodes in the naive physics representation. In the absence of semantic knowledge, the heuristic allows a way to order the search when qualitative values are propagated.

Each domain equation is encoded in a frame representation that specifies an executable equation, the equation's name, a naive representation parent (if applicable), and relevant semantic information. The semantic information may be an explicit statement of a constraint or a pointer to another knowledge source. The influence links to a node from its independent variables are obtained by a symbolic series expansion of each variable specified in the equation. By this means, a link is defined as a positive or negative influence. Heuristics are used to specify if a link is a primary or secondary influence.

The naive physics representation provides an integration and interpretation of the domain equations. The set of nodes and links define a structure in which qualitative simulation can be accomplished. When ambiguities are encountered, exact values based on domain equations can be derived.

Heuristic knowledge can be justified with mathematical knowledge. Equations can be solved qualitatively. This may be particularly

applicable to a training environment. A novice controller might ask a question like "What happens when the aircraft weight decreases?" The decreased value of weight can be propagated through the network. The reasoning process also operates in a 'bottom up' fashion. A new equation or algorithm can be represented in the existing structure and interpreted in terms of the existing heuristic knowledge base.

The research has spawned a number of interesting problems which should be addressed in future research. The qualitative reasoning approach developed in this work is by no means complete and can be expanded in several directions.

First, while I have concentrated on reasoning about the steady state behavior of dynamic systems, the approach may be equally applicable to transient behavior. The naive physics representation could be expanded to show integrable relationships between variables. For example, velocity is the integral of acceleration. A symbolic transfer function could be generated based on the domain equation interfaces to the naive representation and the transient behavior computed. This would be useful for in reasoning about time-elapsed behavior in mechanical systems. Heuristics from linear control theory could be included. For instance, the time constant of a linear system defines the time required to change states. In the context of an aircraft changing its velocity, the time constant can be on the order of ten seconds.

The theory could be extended to deeper levels of knowledge. For instance, I limited the equation for lift at the angle-of-attack. But, angle-of-attack is a control variable dependent on the pilot's 'stick' position. A complete mathematical description of aircraft performance

may be possible.

I have limited the naive representation to Newtonian mechanics as applied to one-dimensional motion. I have not discovered a way to reason about goal interactions involving goals of different types. This work used the 'meta-plan' that it was better to be safe than efficient. But, the definition of 'safe' and 'efficient' are not precise enough to make the meta-plan infallible. For instance, a human controller will take into account the perceived desires of an aircraft pilot in his plans. This suggests that the controller not only uses naive theories of motion, but naive theories of human psychology, naive economics, etc.

I have not completely explored advice-initiated learning. Advice indicates some novelty in the present problem solving context. The computer can apply the qualitative reasoning approach to another expert's advice, but this makes the assumption that there is sufficient heuristic knowledge of strategies and tactics to create a new representation. I have not investigated how the computer might operationalize or generalize an interpreted plan. It seems plausible that the computer could summarize the results of a reasoning task as a 'rough' heuristic which would then be subject to criticism in future problem solving contexts.

APPENDIX A

AIRCRAFT PERFORMANCE EQUATIONS

Subsonic aircraft performance can be computed from airframe and engine characteristics and atmospheric conditions. These data are readily available [63]. The equations for the basic optimum flight conditions of level flight are derived. It is important to note that the same knowledge (embedded in tabular form) is used by the aircraft pilot to plan his flight profile. These equations yield data which compare favorably with the performance data contained in the DC10 pilot performance handbook [23]. The following units are used: horizontal velocity (knots true airspeed or KTAS), vertical velocity (ft/min), forces (pounds), and linear measurements (ft).

1.0. Thrust Required

An aircraft in steady state level flight is at an equilibrium condition such that lift equals weight and drag equals the required thrust. If the available thrust exceeds the drag, then the aircraft may fly faster or climb higher. Lift and drag are defined as follows:

$$L = C_L \, q \, S \tag{A.1}$$

$$D = C_D \, q \, S \tag{A.2}$$

where C_L is a nondimensional lift coefficient that relates lift to the aircraft's angle of attack, q is called dynamic pressure ($1/2 \, \rho \, V^2$), and S is the aircraft lift surface area (well approximated by the wing

area). Drag consists of two components: parasitic drag and induced drag (see Fig. 4, p. 29). Parasitic drag, D_p, and induced drag, D_i, are defined as follows:

$$D_p = C_{D_p} q S \tag{A.3}$$

$$D_i = C_{D_i} q S \tag{A.4}$$

where C_{D_p} is a constant and the quantity (C_{D_p} S) is called the equivalent surface area (f) and

$$C_{D_i} = \frac{C_L^2}{\pi AR} \tag{A.5}$$

where AR is the aspect ratio (wing-length^2/S). Note that from the equilibrium condition and (A.1):

$$C_L = \frac{L}{1/2\ \rho\ V^2\ S} = \frac{W}{1/2\ \rho\ V^2\ S} \tag{A.6}$$

The total aircraft drag is the summation of parasitic drag (proportional to V^2) and induced drag (inversely proportional to V^2) as shown below:

$$D_{req} = \frac{f\ V^2\ \sigma}{295} + \frac{94}{\sigma e} \left|\frac{w}{b}\right|^2 V^{-2} \tag{A.7}$$

In the above equation, f is the equivalent surface area and e is the

Oswald efficiency factor (determined from wind tunnel tests on the wing shape). Since at equilibrium thrust equals drag, the above equation yields the required thrust to maintain a velocity V for a given aircraft configuration (i.e., a given W, S, b).

2.0. Thrust Available

The available thrust is given in terms of the engine specifications. It should be noted that the specified thrust (calibrated on the ground under ideal conditions) is degraded at altitude by pressure and temperature effects. This is because the air mass flowing into the engine is reduced. This degradation is compensated by 'ram effect', the increased air flow resulting from the velocity of the aircraft. For the purposes of this research, it is assumed that the available thrust is constant for the range of cruise velocities usually encountered in the high altitude enroute air traffic control problem. For jet engines, the specific fuel consumption (e.g., fuel-flow rate/thrust) is constant in this velocity range.

3.0. Optimum Flight Conditions

The thrust required and thrust available curves are sufficient to compute the following: maximum endurance airspeed, maximum range airspeed, maximum airspeed, best angle rate-of-climb, maximum rate-of-climb, optimum cruise altitude, and service ceiling.

3.1. Maximum Endurance Airspeed

For a given weight and altitude, the maximum endurance airspeed occurs when the required thrust is minimized. Intuitively, this is also

the point where the lift to drag ratio is maximized. The ratio of lift to drag is equivalent to the ratio of C_L to C_D. Thus, maximum endurance velocity can be computed by differentiating the lift to drag ratio, solving for the maximum lift to drag condition, and then solving for V in (A.7). These calculations yield:

$$\left|\frac{L}{D}\right|_{max} = 0.886\ b\ \left|\frac{e}{f}\right|^{1/2} \tag{A.8}$$

$$V_{L/D_{max}} = \frac{12.9}{(fe)^{1/4}}\ \left|\frac{W}{\sigma b}\right|^{1/2} \tag{A.9}$$

Also, the required thrust and power at L/D_{max} is:

$$T_{L/D_{max}} = 1.132\ \frac{W}{b}\ \left|\frac{e}{f}\right|^{1/2} \tag{A.10}$$

$$P_{L/D_{max}} = \frac{T_{L/D_{max}}\ V_{L/D_{max}}}{375} \tag{A.11}$$

3.2. Maximum Velocity

The maximum velocity occurs when the available thrust equals the required thrust for a given aircraft configuration. Thus V_{max} is easily found by solving (A.7) for V when T_R equals T_A.

$$V_{max} = \left|\frac{T_A + \left|T_A^2 - 1.274\ \frac{f}{e}\ \left|\frac{W}{b}\right|^2\right|^{1/2}}{\frac{\sigma f}{147.5}}\right|^{1/2} \tag{A.12}$$

3.3. Maximum Range Airspeed

Since for a jet engine, thrust is proportional to fuel flow rate, the speed for maximum range for a given aircraft configuration lies between the maximum endurance airspeed and the maximum airspeed. Graphically, this point is the tangent on the T_R curve of a line passing through the origin. The maximum range airspeed is the airspeed where V (fuel-reserves/fuel-flow-rate) is maximized. Since specific fuel consumption, c, is constant with respect to velocity, then the fuel flow rate for a given velocity is c T_R. Thus range is given as:

$$\text{Range} = V\, W_{fuel}/(c\, T_r) \tag{A.13}$$

and the maximum range airspeed is found by differentiating the range equation with respect to V. Thus,

$$V_{max\text{-}range} = \frac{17}{(fe)^{1/4}} \left|\frac{W}{\sigma\, b}\right|^{1/2} \tag{A.14}$$

3.4. Best Angle of Climb

The best angle of climb occurs at $V_{L/D_{max}}$.

$$R/C = 101.27\, \frac{T_A - T_R}{W}\, V \tag{A.15}$$

The best angle of climb then is:

$$Ang = \sin^{-1} \left| \frac{T_A - T_R}{W} \right| \tag{A.16}$$

Note that the minimum sink angle also occurs at $V_{L/D_{max}}$.

3.5. Maximum Rate of Climb

The maximum rate of climb is found by nondimensionalizing eq. (A.15) and applying the calculus of variations.

Let

$$R/C = 101.27 \frac{T_{L/D_{max}} V_{L/D_{max}} \frac{\Delta T}{T_{L/D_{max}}} \frac{V}{V_{L/D_{max}}}}{W} \tag{A.17}$$

$$x = \frac{V}{V_{L/D_{max}}} \tag{A.18}$$

$$y = x^2 + x^{-2} \tag{A.19}$$

$$y_1 = \frac{2\ T_A}{T_{L/D_{max}}} \tag{A.20}$$

Observing that,

$$\frac{2\ T_R}{T_{L/D_{max}}} = \left| \frac{V}{V_{L/D_{max}}} \right|^2 + \left| \frac{V}{V_{L/D_{max}}} \right|^{-2} \tag{A.21}$$

Then the maximum rate of climb occurs when x Δy is maximized.

$$x\,\Delta y = x\,y_1 - x^3 - x^{-1} \tag{A.22}$$

$$\frac{d(x\,\Delta y)}{dx} = y_1 - 3\,x^2 + x^{-2} \tag{A.23}$$

Thus, the maximum rate of climb occurs when

$$x = \left| \frac{y_1 + \left| y_1^2 + 12 \right|^{1/2}}{6} \right|^{1/2} \tag{A.24}$$

3.6. Optimum Altitude

The optimum altitude for a given aircraft configuration is found by differentiating (A.7) with respect to altitude, h, where the density ratio, σ, is a function of h. This yields

$$h = dr^{-1} \left| \frac{27735\,W^2}{f\,e\,b^2\,V^4} \right|^{1/2} \tag{A.25}$$

where dr^{-1} is the inverse density ratio function.

3.7. Service Ceiling

The service ceiling for a given aircraft configuration is the altitude where the maximum rate of climb is 100 ft/min. This altitude is found by solving eq. (A.15) for h when R/C equals 100 ft/min.

$$h = dr^{-1} \left| \frac{A + \left| A^2 - 4\,B\,C \right|^{1/2}}{2\,B} \right| \tag{A.26}$$

where

$$A = T_A - \frac{R/C\ W}{101.27} V^{-1} \qquad (A.27)$$

$$B = \frac{f}{295} V^2 \qquad (A.28)$$

$$C = \frac{24}{e} \left|\frac{W}{b}\right|^2 V^{-2} \qquad (A.29)$$

APPENDIX B

CONFLICT FILTER EQUATIONS

Let (x_1, y_1) and (v_1, θ_1) be the position and velocity of aircraft$_1$ at time t_1, and (x_2, y_2) and (v_2, θ_2) be the position and velocity of aircraft$_2$ at time t_2. Then, define the following terms.

$$v_{1x} = v_1 \cos \theta_1 \tag{B.1}$$

$$v_{1y} = v_1 \sin \theta_1 \tag{B.2}$$

$$v_{2x} = v_2 \cos \theta_2 \tag{B.3}$$

$$v_{2y} = v_2 \sin \theta_2 \tag{B.4}$$

$$x_R = x_2 - x_1 \tag{B.5}$$

$$y_R = y_2 - y_1 \tag{B.6}$$

$$V_{Rx} = v_{2x} - v_{1x} \tag{B.7}$$

$$V_{Ry} = v_{2y} - v_{1y} \tag{B.8}$$

$$A = v_{1x} v_{2x} + v_{1y} v_{2y} \tag{B.9}$$

$$B = V_{Rx} x_R + V_{Ry} y_R \tag{B.10}$$

$$C = v_{1x} v_{2y} - v_{2x} v_{1y} \tag{B.11}$$

$$D = V_{Rx} y_R - V_{Ry} x_R \tag{B.12}$$

then the time of closest approach is given by:

$$\frac{(v_1^2 - A)\ t_1 + (v_2^2 - A)\ t_2 - B}{V_{Rx}^2 + V_{Ry}^2} \tag{B.13}$$

and the distance of closest approach is given by

$$\frac{C\ (t_2 - t_1) + D}{V_{Rx}^2 + V_{Ry}^2} \tag{B.14}$$

These results can be derived by expressing Δx and Δy, the x and y components, respectively, of the separation between the aircraft as a function of time t:

$$\Delta x = (x_R + v_{1x}\ t_1 - v_{2x}\ t_2) + V_{Rx}\ t \tag{B.15}$$

$$\Delta y = (y_R + v_{1y}\ t_1 - v_{2y}\ t_2) + V_{Ry}\ t \tag{B.16}$$

The point of closest approach occurs at that time which minimizes $(\Delta x^2 + \Delta y^2)$, the square of the distance between the aircraft. This time can be readily obtained by setting $\frac{d}{dt}[\Delta x^2 + \Delta y^2] = 0$ and solving for t. The distance of closest approach can be obtained by substituting the value of t into the expression $[\Delta x^2 + \Delta y^2]^{1/2}$. However, a great deal of algebraic computation can be avoided by eliminating t from the pair of equations for Δx and Δy. The resulting linear equation in the variables Δx and Δy represents the trajectory traversed by one aircraft relative to the other. The distance from this line to their origin is the distance of closest approach between the two aircraft. By putting this equation into normal form (i.e., $\alpha\ \Delta x + \beta\ \Delta y + \delta = 0$, where $(\alpha^2 + \beta^2 = 1)$, the distance from the origin is simply the value of the constant the coefficient, δ.

APPENDIX C

SUMMARY OF ENROUTE AIR TRAFFIC CONTROL KNOWLEDGE

The enroute ATC domain knowledge is summarized. The knowledge was obtained through discussions with air traffic controllers and a review of the available ATC literature. This knowledge has been successfully used to resolve conflicts in Chicago ARTCC DYSIM training problems.

Controllers resolve conflicts in a reactive manner. That is, once a conflict is predicted, a resolution command is chosen based on stereotypical knowledge of the conflict situation. This knowledge is obtained through extensive on-the-job training. Experienced controllers evolve a particular style for controlling aircraft. Significantly, controllers comprehend the plans of other controllers.

The strategic knowledge consists of conflict prediction (goal formation), designation of right-of-way and subject aircraft, and preferred commands. The conflict prediction problem has been discussed [Appendix B]. Rules for choosing right-of-way and subject aircraft are defined. The preferred commands are functions of the conflict type (e.g., head-on, crossing) the aircraft intent, and the aircraft state vector.

The tactical knowledge consists of domain rules and constraints. A primitive vocabulary of 21 commands is defined. This knowledge was acquired from controllers during visits to the Chicago ARTCC and the FAA Academy. The AERA algorithms for fuel efficient aircraft maneuvers are

included.

1.0. Strategic Knowledge

Given a list of aircraft conflicts, the purposes of the conflict resolution strategies are to:

1. Define a right-of-way (ROW) aircraft and an aircraft subject to resolution commands, and
2. Define the preferred order in which resolution commands will be attempted.

A set of meta-rules that determine the ROW and subject aircraft for a given conflict pair are defined. The determination is based on the individual aircraft state vectors and intent. The state vector consists of the aircraft's position and velocity vectors. Aircraft intent refers to the flight plan. At the time of the conflict, an aircraft can have one of five intents:

1. level enroute
2. climbing enroute
3. descending enroute
4. departure
5. arrival

For example, the Joliet sector of the Chicago area includes O'Hare. An aircraft entering Joliet from O'Hare has the departure intent. An aircraft flying from Boston to Los Angeles through the Joliet airspace has the level enroute intent if it has not requested or been instructed to transition altitude.

The meta-rules for the determination of right-of-way and subject aircraft are listed below.

1. If the involved aircraft have the same intent, then the subject aircraft is the aircraft farthest from the conflict point.

2. If the involved aircraft have the same intent, then the subject aircraft is the aircraft at the lower altitude.

3. An aircraft in level flight has the right-of-way over an aircraft transitioning altitude.

4. A climbing aircraft has right-of-way over a descending aircraft.

5. An enroute-descending aircraft has the right-of-way over an arrival aircraft.

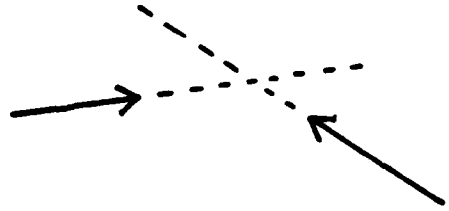
6. An enroute-climbing aircraft has the right-of-way over a departure aircraft.

7. If no other rules apply, designate one aircraft as the subject and the other as having the right-of-way.

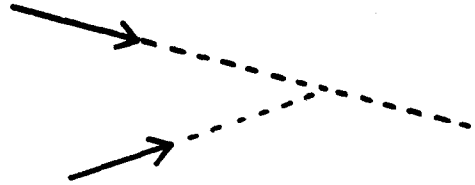
The resolution command preferences were defined as a function of aircraft intent, conflict type, and control procedures. Four types of conflicts are illustrated below.

1. Head-on

2. Crossing



3. Merging



4. Overtake

There are two fundamental control procedures: separation of aircraft from airspace and separation of aircraft from aircraft. The first is the most prevalent. It is referred to as positive control and is the standard procedure when conflicts are predicted well in advance and the controller is directing aircraft with the radar display. The controller can see the affect of the commands he issues. The procedure requires the controller to place a protection circle around the right-of-way aircraft and issue a resolution command that will keep the subject aircraft from violating the protected airspace in a fuel efficient manner. The second procedure results when an aircraft suddenly appears on the screen, the controller does not predict the conflict, or the radar screen malfunctions. The controller relies heavily on altitude changes and keeping aircraft on their filed routings. We have only included the stereotypical knowledge as it applies to a positive control.

In Figs. 26-29 (p. 115-118), the preferred command sequences where positive control is exercised are shown. The columns indicate aircraft intent and the rows indicate decreasing preference of command. We define the additional meta- rule:

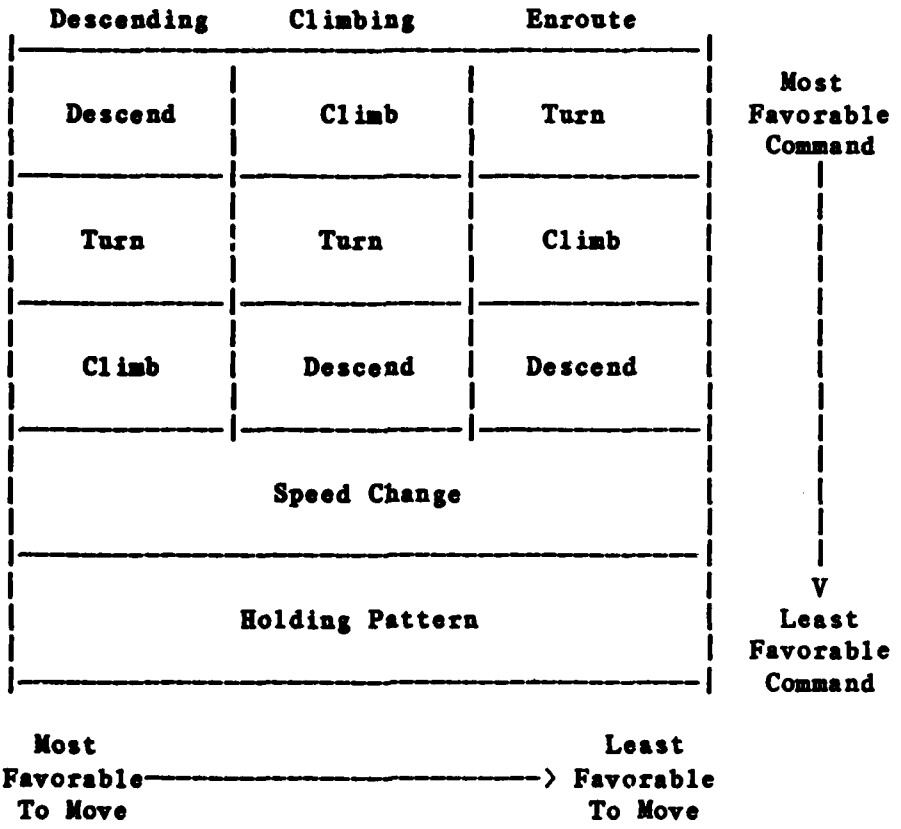
1. The right-of-way aircraft and the subject aircraft 'flip-flop' when it is impossible to implement a preferred command.

As an example consider two aircraft, UA86 and AA75, involved in a crossing conflict. Let UA86 have a level enroute intent and AA75 have a departure intent. The control procedure is positive control. The preferred sequence of commands would be:

1. Descend AA75
2. Turn UA86
3. Turn AA75
4. Climb UA86

Crossing Conflict

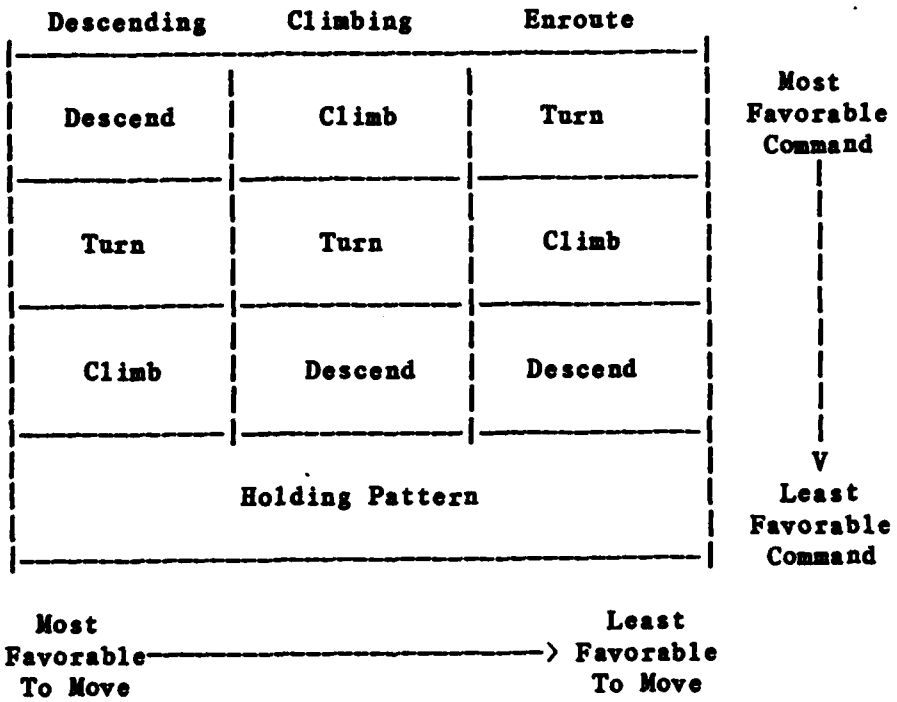
Aircraft State Vector



Crossing Conflict Command Preferences
Figure 26

Head-on Conflict

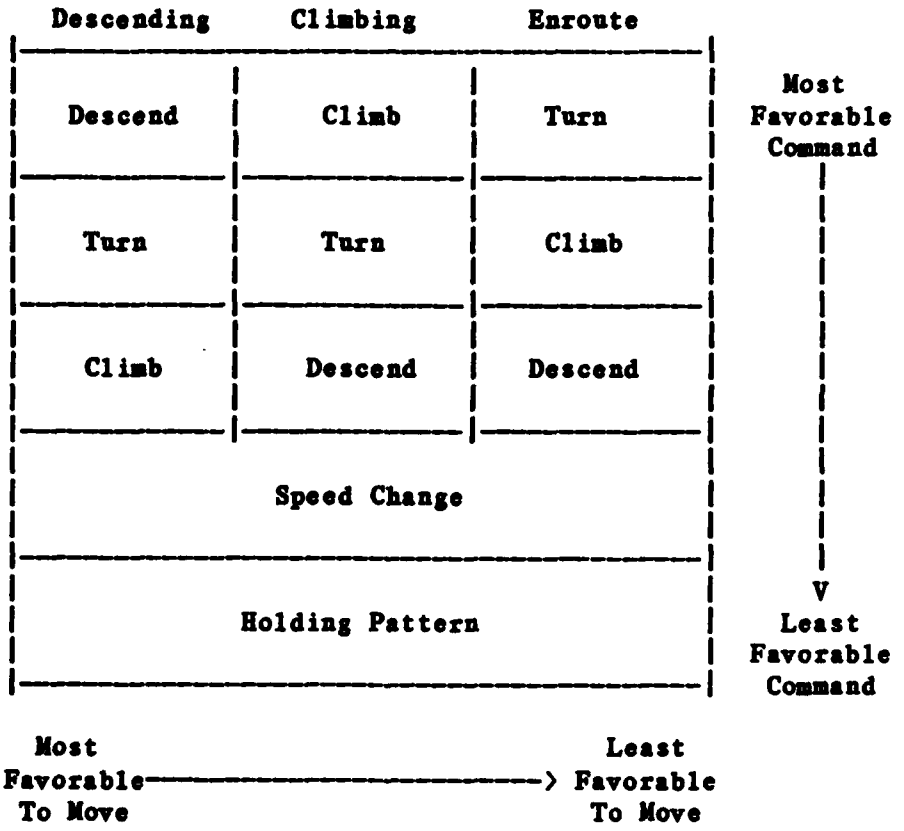
Aircraft State Vector



Head-on Conflict Command Preferences
Figure 27

Merging Conflict

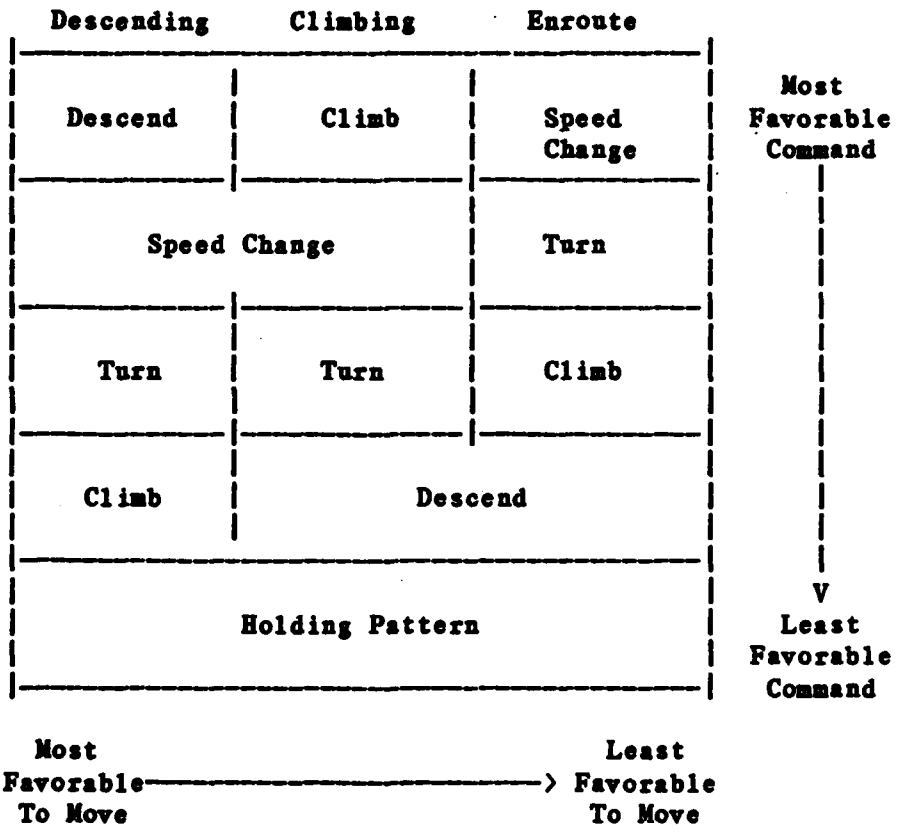
Aircraft State Vector



Merging Conflict Command Preferences
Figure 28

Overtake Conflict

Aircraft State Vector



Overtake Conflict Command Preferences
Figure 29

5. Climb AA75
6. Descend UA86
7. Speed Change AA75
8. Speed Change UA86
9. Hold AA75
10. Hold UA86

2.0. Tactical Knowledge

The tactical knowledge we discussed consists of domain rules and constraints, AERA equations and algorithms and a primitive command vocabulary.

2.1. Rules and Constraints

We have defined 36 rules to implement climb and descend commands. One typical rule is listed below.

If: (1) The sbjct is level,
(2) Altitude (right-of-way) < altitude (subject),
(3) Conflict type is merging,
(4) Altitude (subject) < 29,000 feet, and
(5) Conflict point is a fix

Th: (1) Descend the subject to altitude (ROW) - 1000
(2) Use the command "DESCEND AND MAINTAIN (altitude)"

Constraint relationships are well defined in the Air Traffic Control Handbook [3]. Some examples include:

1. Eastbound aircraft below 29,000 feet fly even cardinal altitudes.

2. Westbound aircraft below 29,000 feet fly odd cardinal altitudes.

3. If an aircraft is above 29,000, speed adjustments must be approved by the pilot.

4. Do not lower an aircraft's speed below 250 knots.

2.2. AERA Algorithms

We have implemented the AERA equations and algorithms for aircraft vectoring [35]. The equations are functions of the involved aircraft state vectors. They include:

1. Commitment Point. An equation used to determine the latest point along the subject aircraft's route where it can use a specified angle to resolve the conflict.

2. Wedge of Prohibited Angles. Several equations to determine the range of possible turn angles given the point the subject aircraft is to turn off its route.

3. Return Point. An iterative algorithm used to determine when an aircraft can turn back to its route. The algorithm has embedded in it the heuristic that no delay is incurred when an aircraft flies parallel to its intended route. Hence, if an aircraft can not yet be turned back to its route the algorithm seeks to turn the aircraft on a parallel course to its original route.

2.3. Primitive Vocabulary

Part of a controller's style is the manner in which he issues commands. 21 basic commands are defined.

1. Altitude Commands

   a. To indicate pilot should maintain an altitude

      (1). MAINTAIN (Altitude)

      (2). MAINTAIN (Altitude) UNTIL (Time) OR PAST (Fix) OR (Number of miles or minutes) PAST (fix)

      (3). CROSS (Fix) OR INTERCEPT (Route) AT OR ABOVE (Altitude)

   b. Instruct pilot to climb/descend

      (1). CLIMB/DESCEND AND MAINTAIN (Altitude)

      (2). CLIMB/DESCEND AND MAINTAIN (Altitude) WHEN ESTABLISHED AT LEAST (number of miles or minutes) PAST (fix) ON THE (specified) RADIAL

(3). CLIMB/DESCEND TO REACH (Altitude) AT (time) or (fix) OR A POINT (number of miles) MILES (direction) OF (name of DME NAVAID)

c. Specify altitude over a specified fix

(1). CROSS (Fix) AT (Altitude)

(2). CROSS (Fix) AT OR ABOVE/BELOW (altitude)

d. If possible, indicate at pilot's discretion

(1). CLIMB/DESCEND AT PILOT'S DISCRETION

e. Altitude assignment with more than one altitude

(1). MAINTAIN (Altitude) THROUGH (Altitude)

2. Vectoring Aircraft

a. Commands to initiate vectoring

(1). TURN RIGHT/LEFT HEADING (degrees)

(2). FLY HEADING (degrees)

(3). FLY PRESENT HEADING

(4). DEPART (fix) HEADING (degrees)

b. If possible, advise pilot of the purpose

(1). FOR VECTOR TO (fix or airway)

(2). VECTOR FOR SPACING

(3). EXPECT TO RESUME (route, etc.)

3. Holding Commands

a. Commands

(1). HOLD (direction) OF (fix) ON (specified radial, course, bearing airway, or jet route) (number of miles) MILE LEG (and if left turns) LEFT TURNS

4. Speed Adjustments

a. To maintain, increase/decrease, or avoid exceeding

(1). SAY AIRSPEED, SAY MACH MANNER

(2). INCREASE/REDUCE SPEED TO (number of knots) OR TO MACH (Mach number)

(3). DO NOT EXCEED (speed or Mach number)

b. If pilot concurrence is required

(1). IF PRACTICAL, REDUCE SPEED TO (number of knots) (number) KNOTS

c. If coupled with a descent, make order explicit

(1). REDUCE SPEED TO (specified speed) OR (number of knots) KNOTS, THEN, DESCEND AND MAINTAIN (altitude)

(2). DESCEND AND MAINTAIN (altitude), THEN, REDUCE SPEED TO (specified speed) OR (number of knots) KNOTS.

d. Limitations

(1). IF PRACTICAL, MAINTAIN (specified speed)

(2). IF PRACTICAL, INCREASE/REDUCE SPEED (specified knots)

APPENDIX D

AUTOMOBILE DOMAIN EQUATIONS

Propulsive Force

$$F_{RW} = T_{RW} / r \quad \text{(D.1)}$$

$$T_{RW} = T_{CO}\ gr\ \mu \quad \text{(D.2)}$$

$$T_{CO} = TR\ T_{ENG} \quad \text{(D.3)}$$

$$T_{ENG} = N_E^2 / K_I^2 \quad \text{(D.4)}$$

$$TR = f(SR) \simeq -1.67\ SR + 2.0 \quad \text{(D.5)}$$

$$K_I^2 = f(SR) \simeq 12.0\text{–}16.0\ SR \quad \text{(D.6)}$$

$$SR = N_{CO} / N_{ENG} \quad \text{(D.7)}$$

$$N_{CO} = \frac{60\ v\ gr}{2\ \pi\ r} \quad \text{(D.8)}$$

where

F_{RW} = rear wheel force
R_{RW} = driveshaft torque
T_{CO} = torque convertor output torque
gr = transmission gear shift ratio
μ = transmission efficiency factor
T_{ENG} = engine torque
TR = torque ratio
SR = speed ratio
K_I = input size
N_E = engine speed
N_{CO} = driveshaft speed
v = automobile velocity
r = tire velocity

Resistive Forces

$$D = C_D \rho v^2 A \qquad (D.9)$$

$$TF = C_F W \qquad (D.10)$$

$$F_W = W \sin \theta \qquad (D.11)$$

where,

F_W = force due to weight
W = weight
C_D = drag coefficient
C_F = friction coefficient
A = frontal area
ρ = air density ratio
θ = climb angle
D = drag
TF = Tire force

REFERENCES

[1] Aikins, J. (1980). Prototypes and Production Rules: Knowledge and Representation for Consultation. Ph.D. Dissertation. Stanford University. STAN-CS-80-814.

[2] Airman's Information Manual. (1982). Superintendent of Documents. Government Printing Office. Washington, D.C.

[3] Air Traffic Control Handbook. (1982). Federal Aviation Administration. 7110.65C.

[4] Anderson J. (1976). Language, Memory, and Thought. Lawrence Erlbaum Associates, Publishers. Hillsdale, N. J.

[5] Automotive Handbook. (1982). U. Adler (Ed.). Society of Automotive Engineers.

[6] Beachley, N. and Frank, A. (1974). Increased Fuel Economy in Transportation Systems by Use of Fuel Management. Department of Transportation. DOT-OS-30112.

[7] Bobrow, D., Kaplan, R., Kay, M., Norman, D., Thompson, H., and Winograd, T. (1977). GUS, A Frame-driven Dialog System. Artificial Intelligence. 8:155-173.

[8] Bobrow, D. and Winograd, T. (1977). An Overview of KRL, A Knowledge Representation Language. Cognitive Science. 1:3-46.

[9] Buchanan, B. and Mitchell, T. (1978). Model Directed Learning of Production Rules. in Pattern-Directed Inference Systems. D. Waterman and F. Hayes-Roth (Eds.). Academic Press, New York City.

[10] Bundy, A. (1978). Will It Reach The Top? Prediction in the Mechanics World. Artificial Intelligence. 10:129-146.

[11] Burgess, J. (1982). FAA Chief Circulating Multi-billion Dollar Air Traffic Control Plan. in the Washington Post. January 1, 1982.

[12] Chandrasekaran, B. and Mittal, S. (1982). Deep Versus Compiled Knowledge Approaches to Diagnostic Problem-Solving. Proceedings of the National Conference on Artificial Intelligence. pp. 349-354.

[13] Charniak, E. (1981). A Common Representation for Problem-Solving and Language Comprehension Information. Artificial Intelligence. 16:225-255.

[14] Charniak, E. (1978). On the Use of Framed Knowledge in Language Comprehension. Artificial Intelligence. 11:225-265.

[15] Charniak, E., Riesbeck, C., and McDermott, D. (1981). Artificial Intelligence Programming. Lawerence Erlbaum Associates, Publishers. Hillsdale, N.J.

[16] Chen, C. (1970). Introduction to Linear System Theory. Holt, Rinehart, and Winston, Publishers. New York, N.Y.

[17] Chien, R., Cross, S., Keller, S., Ho, W. (1983). Artificial Intelligence and Human Error Prevention Study. University of Illinois. CSL-T-117.

[18] Clancey, W. (1981). The Epistemology of a Rule-based Expert System: A Framework for Explanation. Stanford University. STAN-CS-81-896.

[19] Davis, R. (1977). Interactive Transfer of Expertise: Acquisition of New Inference Rules. Fifth International Joint Conference on Artificial Intelligence. pp. 321-328.

[20] Davis, R. (1980). Meta Rules: Reasoning About Control. Artificial Intelligence. 15:279-322.

[21] Davis, R. (1976). Use of Meta Level Knowledge in the Construction, Maintenance, and Use of Large Knowledge Bases. Ph.D. Dissertation. Stanford University. STAN-CS-76-552. (AD-76-25988).

[22] Davis, R. and King, J. (1977). An Overview of Production Systems. in Machine Intelligence 8. E. Elcock and D. Michie (Eds.). Ellis Harwood Publishers, Ltd. Chichester, England. pp. 300-332.

[23] DC-10-10 Aircraft Performance Manual. (1979). United Airlines.

[24] DeJong, G. (1982). Automatic Schema Acquisition in a Natural Language Environment. Proceedings of the National Conference on Artificial Intelligence. pp. 410-413.

[25] de Kleer, J. (1977). Qualitative and Quantitative Reasoning in Classical Mechanics. Fifth International Joint Conference on Artificial Intelligence. pp. 299-304.

[26] de Kleer, J. (1979). Causal and Teleological Reasoning in Circuit Recognition. Ph.D. Dissertation. Massachusetts Institute of Technology. MIT-TR-529.

[27] de Kleer, J. and Brown, J. (1982). Foundations of Envisioning. Proceedings of the National Conference on Artificial Intelligence. pp. 434-437.

[28] Duda, R., et al. Development of the PROSPECTOR Consultation System for Mineral Exploration. Stanford Research Institute, Publishers. Final Reports on Projects 5821 and 6415.

[29] Enroute Instructional Program Guide. (1981). Federal Aviation Administration. Air Traffic Control Academy. Oklahoma City, Oklahoma. EP-12-0-1C.

[30] Feigenbaum, E., et al. (1971). On Generality and Problem Solving: A Case Study Using the DENDRAL Program. in Machine Intelligence 6. B. Metzer and D. Michie (Eds.). American Elsevier, Publishers. Chichester, England. pp. 165-190.

[31] Forbus, K. (1982). Qualitative Process Theory. Massachusetts Institute of Technology. MIT-AI-TR-664.

[32] Forbus, K. and Stevens, A. (1981). Using Qualitative Simulation to Generate Explanations. Bolt, Beranek, and Newman, Inc. Report No. 4490.

[33] Hayes, P. (1979). The Naive Physics Manifesto. in Expert Systems in the Micro Electronic Age. D. Michie (Ed.). Edinburgh University Press. Edinburgh. pp. 242-270.

[34] Hopkins, V. (1975). The Controller Versus Automation. in A Survey of Modern Air Traffic Control. A. Benoit (Ed.). AGARD-AC-209. pp. 45-60. (AD-A014512).

[35] Kingsbury, J. (1980). Basic Horizontal Resolution Equations for AERA. Mitre Corp. WP-80W00473.

[36] Knox, C., et al. (1980). Preliminary Test Results of a Flight Management Algorithm for Fuel Conservative Descents in a Time Based Metered Traffic Environment. National Aeronautics and Space Administration. NASA-TM-80194.

[37] Kuipers, B. (1982). Getting the Envisionment Right. Proceedings of the National Conference on Artificial Intelligence. pp. 209-212.

[38] Lachman, R., Lachman, J., and Butterfield, E. (1979). Cognitive Psychology and Information Processing. Lawerence Erlbaum Associates, Publishers. Hillsdale, N.J.

[39] Lerner, E. (1982). Automating U.S. Air Lanes: A Review. IEEE Spectrum. 19:46-51.

[40] McCarthy, J. (1968). Programs with Common Sense. in Semantic Information Processing. M. Minsky (Ed.). pp. 403-409. The MIT Press. Cambridge, Mass.

[41] McCloskey, M. (1983). Naive Theories of Motion. in Mental Models. D. Gentner and A. Stevens (Eds.). Lawerence Erlbaum Associates, Publishers. Hillsdale, N.J.

[42] McDermott, J. (1980). R1: A Rule-Based Configurer of Computer Systems. Carnegie-Mellon University. CMU-CS-80-119.

[43] McDermott, J. and Forgy, C. (1978). Production System Conflict Resolution Strategies. in Pattern-Directed Inference Systems. pp. 177-202. Academic Press. New York, N.Y.

[44] Minsky, M. (1975). A Framework for Representing Knowledge. in The Psychology of Computer Vision. P. Winston (Ed.). McGraw-Hill Book Co., New York, N.Y.

[45] Mitchell, T. (1978). Version Spaces: An Approach to Concept Learning. Ph.D. Dissertation. Stanford University. STAN-CS-78-711.

[46] Mostow, D. (1981). Mechanical Transformation of Task Heuristics into Operational Procedures. Ph.D. Dissertation. Carnegie-Mellon University. CMU-CS-81-113. (AD-A104-674).

[47] Nelson, K. (1976). Airbreathing Engine Notes. USAF Test Pilot School. Edwards AFB.

[48] Pople, H. (1979). The Formation of Composite Hypothesis in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning. Sixth International Joint Conference on Artificial Intelligence. pp. 1030-1037.

[49] Post, E. (1943). Formal Reductions of the General Combinatorial Problem. American Journal of Mathematics. 65:197-268.

[50] Perkins, C. and Hage, R. (1967). Airplane Performance Stability and Control. John Wiley and Sons, Inc., New York, N.Y.

[51] Rieger, C. (1975). The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief, and Contextual Language Comprehension. University of Maryland. TM-373.

[52] Roberts, R. and Goldstein, I. (1977). The FRL Primer. Massachusetts Institute of Technology. MIT-AI-TR-408.

[53] Rucker, R. (1980). Preliminary Functional Characterization of Automatic Planning and Control in En Route Air Traffic Control. Mitre Corp. WP-80W00230.

[54] Sacerdoti, E. (1974). Planning in a Hierarchy of Abstraction Spaces. Artificial Intelligence. 5:115-135.

[55] Schank, R. and Abelson, R. (1977). Scripts Plans Goals and Understanding. Lawerence Erlbaum Associates, Publishers. Hillsdale, N.J.

[56] Shortliffe, E., et al. (1975). Computer-based Consultations in Clinical Therapeutics - Explanation and Rule Acquisition Capabilities of the MYCIN System. Computers and Biomedical Research. 8:303-320.

[57] Simon, H. (1969). The Sciences of the Artificial. The MIT Press. Cambridge, Mass.

[58] Soloway, E. (1978). Learning = Interpretation + Generalization: A Case Study in Knowledge Driven Learning. Ph.D. Dissertation. University of Massachusetts. (ADA-79-03844).

[59] Stefik, M. (1980). Planning with Constraints. Ph.D. Dissertation. Stanford University. STAN-CS-80-874.

[60] Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L. Hayes-Roth, F., Sacerdoti, E. (1982). The Organization of Expert Systems, A Tutorial. Artificial Intelligence. 18:135-173.

[61] Stengel, R. and Marcus, F. (1976). Energy Management Techniques for Fuel Conservation in Military Aircraft. Analytic Sciences Corp. TASC-TR-545-1. (AD-A013-527).

[62] Sussman, G. (1973). A Computational Model of Skill Acquisition. Ph.D. Dissertation. Massachusetts Institute of Technology. MIT-AI-TR-297.

[63] Taylor, J. (1981). Jane's All The World's Aircraft. Jane's Publishing Co., Ltd. London, England

[64] Wesson, R. (1980). Air Traffic Control: Whither Thou Goest? unpublished technical report.

[65] Wesson, R. (1977). Planning in the World of the Air Traffic Controller. Fifth International Joint Conference on Artificial Intelligence. pp. 473-479.

[66] Wilensky, R. (1983). Planning and Understanding. Addison-Wesley Publishing Co., Inc., Reading, Mass.

[67] Winston, P. (1978). Learning by Creating and Justifying Transfer Frames. Artificial Intelligence. 10:147-172.

VITA

Stephen Edward Cross was born in Washington, D.C. on October 7, 1951. He attended the University of Cincinnati where he earned the B.S. in electrical engineering degree in 1974. He then entered active duty with the Air Force. In 1977, while on active duty, he received the M.S. in electrical engineering degree from the Air Force Institute of Technolgy.

Between 1977 and 1980, Captain Cross attended the USAF Test Pilot School at Edwards AFB, California and served as the avionics flight test manager for the Air Launched Cruise Missle (ALCM) flight test program. He received the Ph. D. degree from the University of Illinois in 1983 and is presently serving as an Assistant Professor of Electrical Engineering at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

He is a member of Tau Beta Pi and Eta Kappa Nu.

END

FILMED

1-84

DTIC